National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

UNIVERSITY OF ALBERTA

Integrated Distributed Intelligent System for Differential Pressure Flowmeter Selection and Sizing

by

Murray Glenn Richard Stevenson

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science

in

Process Control

DEPARTMENT OF CHEMICAL ENGINEERING

EDMONTON, ALBERTA

SPRING 1994

National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-612-11381-7

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR:	Murray Glenn Richard Stevenson
TITLE OF THESIS:	Integrated Distributed Intelligent System for Differential Pressure Flowmeter Selection and Sizing
DEGREE:	Master of Science
YEAR THIS DEGREE GRANTED:	1994

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Murray Stevenson

8704 - 147 Street

Edmonton, Alberta

Canada T5R 0Y2

Dec 17 1993

DATE

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommended to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Integrated Distributed Intelligent System for Differential Pressure Flowmeter Selection and Sizing** submitted by **Murray Glenn Richard Stevenson** in partial fulfillment of the requirements for the degree of **Master of Science in Process Control**.

Dr. Ming Rao, Supervisor

Dr. Roger Toogood

Dr. Reginald Wood

Dec. 3, 1993
DATE

Abstract

The selection of an appropriate flowmeter for a specific application is a difficult task because there is a large decision-making space, numerous considerations and an ill-structured problem. This results in a problem that is not applicable to strictly numerical, algorithmic type solutions. Therefore, artificial intelligence techniques have been considered, which emphasize symbolic reasoning and non-algorithmic solutions.

In order to assist non-experts in the selection of an appropriate flowmeter for a specific application, an integrated coordinated knowledge environment (computer program) has been developed. This environment is based on artificial intelligence and object-oriented program techniques. In particular, the environment employs the concept of an integrated distributed intelligent system (IDIS). An IDIS combines independent specialized software packages into an integrated coordinated environment under the control of a meta-system.

In this thesis, an integrated distributed intelligent system for differential pressure flowmeter selection and sizing (IDISDPFSS) has been developed that assists non-expert users select an appropriate flowmeter based on users' specified requirements. Four flowmeter types can be selected: variable area, V-Cone, flow nozzle and orifice plate. If more than one flowmeter type is applicable, the system will rank them accordingly. Once the selection process is completed, the selected flowmeters can be sized with numerical computational packages integrated into the system. In addition, the system integrates a viscosity coupling system for fluid property prediction.

IDISDPFSS has a user-friendly interface and is constructed so that new knowledge and flowmeter types can be easily added. In order to achieve the coordinated distributed

knowledge environment, different software packages, hardware platforms and operating systems are integrated.

Acknowledgments

I would like to express my appreciation to the following groups of people who as a whole assisted me in completing my degree. The first group is the academic staff here at the University of Alberta. In particular, Dr. Ming Rao, my supervisor, who provided direction, moral support and the opportunity to learn. Also, my chairman Dr. Karl Chuang, and my committee members Dr. Roger Toogood and Dr. Reginald Wood who graciously gave of their time.

I would like to express my gratitude to Professor Haiming Qiu, Dr. Qun Wang and the other members of the Intelligence Engineering Laboratory for their valuable advice and assistance.

The third group of people I would like to thank are those who assisted me with the computing part of my thesis. These people always found the time to steer me towards a solution for a problem I encountered. They are Bob Barton, Dan Bright, Dave Couture, Dr. Yiqiang Shu and Henry Sit.

Another group of people to which I am indebted are those from industry. These people gave freely of their expertise, time and software that formed the bases for this thesis. They are Darrell Barnes, Dr. Pin Du, Ken Leiter, Ken Netzel, Vic Pawluk, Lloyd Takeyasu and Jess Ybanez.

Financial support for this thesis research is from the Natural Sciences and Engineering Research Council of Canada (NSERC).

Table of Contents

Abstract

Acknowledgments

Table of Contents

List of Figures

List of Symbols

		Page
Chapter 1	Introduction	1
1.1	Scope	3
1.2	Organization	3
Chapter 2	Flow Measurement	5
2.1	Objectives and Applications	5
2.2	Techniques and Meter Types	7
2.2.1	Overview	9
2.2.2	Differential pressure flowmeters	10
2.3	Key Factors and Relationships	17
2.4	Problem Definitions	19
Chapter 3	Intelligent Systems	22
3.1	Expert Systems	22
3.2	Coupling Systems	25
3.3	Integrated Distributed Intelligent Systems	27
Chapter 4	Implementation of a meta-system, Meta-COOP	30
4.1	Meta-COOP Architecture	30
4.2	Object-Oriented Programming (OOP)	32

4.2.1 Objects and classes 33

4.2.2 Hierarchical inheritance 35

4.2.3 Encapsulation, message sending and data abstraction 35

4.3 Knowledge Organization and Representation 37

Chapter 5 Individual Software Packages 45

5.1 Variable Area Flowmeters 46

5.2 V-Cone Flowmeters 49

5.3 Flow Nozzle and Orifice Plate Flowmeters 50

5.4 A Coupling System for Selection of Viscosity Models and Prediction 51

5.4.1 Numerical computations 53

5.4.1.1 Peng - Robinson Equation of State (1976) 53

5.4.1.2 Viscosity models 55

Chapter 6 System Implementation 62

6.1 System Integration 62

6.1.1 Hardware, software and UNIX operating system 65

6.1.2 Hardware, software and DOS 67

6.1.3 Remote execution of programs 68

6.2 Building a Knowledge Base 71

6.3 Selection Methodology 72

6.4 Knowledge Base Organization 73

Chapter 7 IDISDPFSS Interface and Demonstration 78

Chapter 8 Conclusions 98

References 101

Appendix A IDISDPFSS Knowledge Bases 105

Appendix B MEM 1.0 Source Code 126

Appendix C Viscosity Coupling System Knowledge in PC Plus 136

List of Figures

		Page
Figure 2.1	General classification scheme for flowmeters	8
Figure 2.2	Classification scheme for differential pressure flowmeters	12
Figure 2.3	Schematic diagram of Meter Equipment Manufacturing, Inc. variable area flowmeter	13
Figure 2.4	Schematic diagram of V-Cone flowmeter	15
Figure 2.5	Schematic diagram of flow nozzle flowmeter	16
Figure 2.6	Schematic diagram of orifice plate	17
Figure 3.1	Components of an expert system	23
Figure 3.2	Non-numerical and numerical information to solve a complex problem	25
Figure 3.3	Illustration of a coupling system	26
Figure 3.4	IDIS architecture	28
Figure 4.1	Meta-COOP architecture	31
Figure 4.2	Example of class and instances	34
Figure 4.3	Example of superclass-subclass inheritance	36
Figure 4.4	Decomposition of a complex engineering problem	39
Figure 4.5	Structure of a unit's header	40
Figure 4.6	Example of an attribute	41
Figure 4.7	Example of a decomposition slot	41
Figure 4.8	Example of a method slot	43
Figure 4.9	Example of a rule slot	44
Figure 5.1	Decision frame knowledge organization	59
Figure 5.2	Schematic diagram of information flow in the coupling system	61
Figure 6.1	Overview of IDISDPFSS integration	64
Figure 6.2	Illustration of meta-system knowledge organization for IDISDPFSS	75

Figure 7.1 IDISDPFSS main menu 78

Figure 7.2 User defined requirements 80

Figure 7.3 Choices pop-up menu 86

Figure 7.4 Selection process window 87

Figure 7.5 Example of explanation feature 88

Figure 7.6 Graphic presentation of selection process conclusions 90

Figure 7.7 Partial list of components available in the viscosity coupling system 91

Figure 7.8 Results from viscosity coupling system 91

Figure 7.9 Illustration of concurrent usage of viscosity coupling system 92

Figure 7.10 Pop-up menu for flowmeter sizing buttons 94

Figure 7.11 MEM 1.0 display window 94

Figure 7.12 V-Cone 3.1 main screen 95

Figure 7.13 FLOWEL 2.0 input screen 95

Figure 7.14 Illustration of concurrent usage of flowmeter sizing programs 96

List of Symbols

A - area available for fluid flow

b - parameter in Peng - Robinson equation of state

D - diameter of pipe

da - density of liquid for MEM flowmeter at operating conditions

df - density of float for MEM flowmeter

dm - density of liquid for MEM flowmeter at 70°F and in standard atmosphere

ft - foot

F - Fahrenheit

hr - hour

k - parameter for differential pressure flowmeter

K - parameter for velocity flowmeters

lb - pound mass

M - molar mass

psia - pounds force per square inch absolute

ΔP - differential pressure

P - pressure (absolute)

Q - volumetric flow rate

Qa - volumetric flow rate for MEM liquid flowmeter at operating conditions

Qm_s - mass flow rate for MEM steam flowmeter at operating conditions

Q_g - volumetric flow rate for MEM gas flowmeter at operating conditions

Q_s - volumetric flow rate for MEM steam flowmeter

Qm - volumetric flow rate for MEM liquid flowmeter at MEM base

Qm_g - volumetric flow rate for MEM gas flowmeter at 70°F and 100 psig

R - universal gas constant

$\mathcal{R}$ - Reynolds number

SG - specific gravity

Sv - specific volume of steam for MEM flowmeter

T - temperature (absolute)

v - specific volume of fluid

Greek Symbols

α - rotational coupling coefficient in Pedersen and Fredenslund equation

μ - absolute viscosity of fluid

θ - shape factor in Ely and Hanley equation

ρ - density of fluid

υ - average fluid velocity flow rate

Ω - shape factor in Ely and Hanley equation

ξ - parameter in Dean and Stiel equation

Superscripts and Subscripts

c - critical property

g - gas phase

m - mixture

o - reference state

r - reduced property

® - registered trademark

TM - trademark

Abbreviations

AGA - American Gas Association

AI - Artificial Intelligence

ASME - American Society of Mechanical Engineers

DOS - Disk Operating System

GPM - Gallons Per Minute

GUI - Graphic User Interface

IBM - International Business Machines

IDIS - Integrated Distributed Intelligent System

!DISDPFSS - Integrated Distributed Intelligent System for Differential Pressure Flowmeter Selection and Sizing

ISO - International Organization for Standardization

MEM - Meter Equipment Manufacturing, Inc.

MS - Microsoft

OOP - Object-Oriented Programming

PC - Personal Computer

P-R - Peng - Robinson equation of state

RAM - Random Access Memory

SCFM - Standard Cubic Feet per Minute

TCP/IP - Transmission Control Protocol/Internet Protocol

Chapter 1

Introduction

Today, in an increasingly competitive marketplace, as process companies strive to lower operating costs and increase productivity, greater emphasis is being placed on process measurements. These measurements are used to monitor and control factors effecting quality and quantity of product. One key measurement is flow.

Flow measurements account for 60% of all measurements in industry (Opie, 1987) and between 50% (Ginesi and Grebe, 1985) to 75% (O'Brien, 1989) of total dollar value spent on instrumentation annually*. Yet, experts claim 60% (Opie, 1987) to over 75% (Meinhold, 1984) of all installed flow measurement devices are inappropriate. The primary cause is improper selection. (Meinhold, 1984, Opie, 1987).

Flowmeter selection is a difficult task (Sovik, 1985; Ginesi, 1991) because a single person must work with a large domain (Rusnak, 1989), consider many related factors (Cheremisinoff, 1979), and deal with an ill-structured problem. Currently, the task is usually performed by a non-expert in the field who must evaluate each application separately and choose from a wide variety of flowmeters (Lomas, 1977). The person's

---

* In 1992, the estimated expenditures on instrumentation in North America were 6 billion dollars Cdn.

final decision requires a balancing of factors (O'Brien, 1989) based on his/her personal knowledge and experience.

In order to assist these non-experts, the techniques of expert systems have been considered. Expert systems are a sub-section of Artificial Intelligence (AI) which has been extensively applied to engineering. Expert systems use computer programming techniques which emphasize symbolic information and non-algorithmic processing to capture and accumulate human experts' knowledge in a specific domain (Buchanan, 1985). In contrast, more traditional computer programming, using languages such as FORTRAN or BASIC, stress numerical and algorithmic procedures.

In review of the literature, two articles have been located that discuss expert systems with regard to flowmeter selection. Lycett and Maudsley (1986) outline considerations to be addressed when developing an expert system for flowmeter selection and sizing, including a plan for developing such a system. However, in subsequent years, no further reference to this system was located. Baker-Counsell (1985) briefly describes a "prototype system" with "much of the detailed work ... yet to be followed up." This system is rule-based and developed using the AI programming language Prolog. Although these systems may not have been implemented or become only partially functional, these articles state a number of requirements for a system: the need to use symbolic information processing, the need for easy modification, and the need to interface symbolic techniques with algorithmic programs. In particular, the system should be able to obtain "Process data such as flowrate, temperature, pressure, and physical properties of the fluid." (Lycett and Maudsley, 1986). Also, the system should not only be capable of selecting a flowmeter but sizing it too.

In an effort to meet the above requirements, an *integrated distributed intelligent system for differential pressure flowmeter selection and sizing* (IDISDPFSS) has been developed. This system is based on the concept of *integrated distributed intelligent system* (IDIS) (Rao, 1991). In the simplest of terms, IDIS is a knowledge integration environment that applies the techniques of artificial intelligence to solve a complex problem by combining independent specialized software packages. The key construct to IDIS is a meta-system. The meta-system coordinates, integrates and communicates with the individual software packages (Rao, 1991). The meta-system, Meta-COOP, has been developed using object-oriented programming (OOP) techniques by the Intelligence Engineering Laboratory in the Department of Chemical Engineering at the University of Alberta. The system is coded in C.

1.1 Scope

IDISDPFSS can select a flowmeter for an application based on the user's supplied requirements. Four types of flowmeters can be selected: variable area, V-Cone, ASME flow nozzle and concentric square edged orifice plate. Applications may be for liquid and/or gas fluids (some particulate matter is acceptable) in full closed circular pipes. The transport and physical properties of complex fluid multi-component mixtures can be calculated using temperature, pressure and composition. The sizing of the flowmeters is provided by three autonomous numerical computing programs.

1.2 Organization

This thesis consists of eight chapters plus three appendices. Chapter 2 introduces flowmetering applications, techniques and considerations as well as problem definitions.

Chapter 3 provides background information on the artificial intelligence techniques applied. Chapter 4 begins by contrasting expert systems with Meta-COOP and then presents basic OOP concepts as a lead into Meta-COOP's knowledge organizational and representational format. Chapter 5 describes the autonomous software packages employed in IDISDPFSS: the numerical flowmeter sizing programs and a coupling system for viscosity prediction. In Chapter 6, integration of the different software packages and the IDISDPFSS knowledge organization are discussed. Chapter 7 illustrates the IDISDPFSS interface and flowmeter sizing packages through a typical selection problem. Chapter 8 contains conclusions. Appendix's A, B and C contain the knowledge bases for IDISDPFSS, the source code for variable area sizing program and the knowledge for the viscosity coupling system, respectively.

Chapter 2

Flow Measurement

In the past, flow measurement was easy; the applications were simple and there were few methods. However, in today's knowledge-intensive industrial manufacturing processes, the measurement technology gets more complicated; the processes are complex and there are many techniques. Flow measurement is a large, highly specialized field. In order to give the reader an appreciation of the complexity of flowmeter selection, a general overview is first presented, including objectives and applications, techniques and meter types, as well as factors to be considered. Then, problem definitions in flowmeter selection and sizing will be discussed.

2.1 Objectives and Applications

Continuous flow measurements account for 83% of all industrial flow measurements (Hasley, 1986) with such three primary functions as indicating (Rusnak, 1989), accounting and controlling (Meinhold, 1984). Indicating measurements convey two types of information: the approximate rate and usually the expected range of flow. As an example, a flowmeter's display may show 0 to 1000 gallons per minute in increments of 100 with possibly the expected normal operating rate between 300 to 700 gallons per minute. In contrast, a meter that displays a single value such as 500 gallons per minute gives no indication of what the expected flow rate should be. An application of indicating flow

measurement is in the monitoring of coolant flow rate in expensive machinery such as compressors. If the flow is near the maximum or minimum rate, this can suggest a problem which perhaps can be corrected before extensive, expensive repairs are required or in the worst case, catastrophic failure.

Accounting measurements sum the quantity (mass or volume) of material over an extended period of time and can be separated into two classes. The first class, namely custody transfer, is concerned with assessing the amount of material received or delivered for satisfying contractual agreements. This class will not be covered because of the stringent requirements imposed by law and the fact that this class accounts for only 4%[†] (Hasley, 1986) of the all measurements. The other class is for internal use by companies to analyze the efficiency or profitability of processes. For example, in the chemical process industry, by measuring the amount of reactant(s) entering and product exiting a reactor, the reactor efficiency can be calculated.

Accurate measurements assist operators in maintaining a plant at optimal operating conditions. On the basis of the output from the flowmeter, a controller, in the case of automatic control, or an operator, in the case of manual control, adjusts a final control element to manipulate process variables. As an example, in combustion processes to ensure efficiency, the flow rates of hydrocarbon fuel and oxygen must be measured in order to maintain a pre-determined ratio. Should the hydrocarbon/oxygen ratio increase, the fuel waste increases due to the incomplete combustion with increased hydrocarbon and the carbon monoxide emission. Should the hydrocarbon/oxygen ratio decrease, the energy waste for heating and pumping the excess oxygen will increase.

[†] This 4% is not included in the 83% stated previously.

Flow measurements are needed in a number of industries over a wide range of operating conditions. Applications vary from the cryogenic production of liquid nitrogen to the high temperature generation of superheated steam; and from the industrial refining of petroleum to the commercial pasteurizing of milk. As a result, flowmeters are required in critical applications for temperatures that vary from -200°C to 500°C, pressures from atmospheric to 3000 psi, pipe diameters from 0.25 inch to 60 inch, and single phase to multi-phase fluid (although no meter can be subjected to all in one service).

2.2 Techniques and Meter Types

Flowmeters have been classified using many different criteria, but the most convenient classification for our purpose is given in Figure 2.1 (Meinhold, 1984). This diagram by no means includes all flowmeter types, but gives an overview of some of the more common measuring techniques and meter types. This section first presents a very brief description of mass, velocity and positive displacement meters. Then a more comprehensive description of the differential pressure flowmeters considered in this thesis is presented.

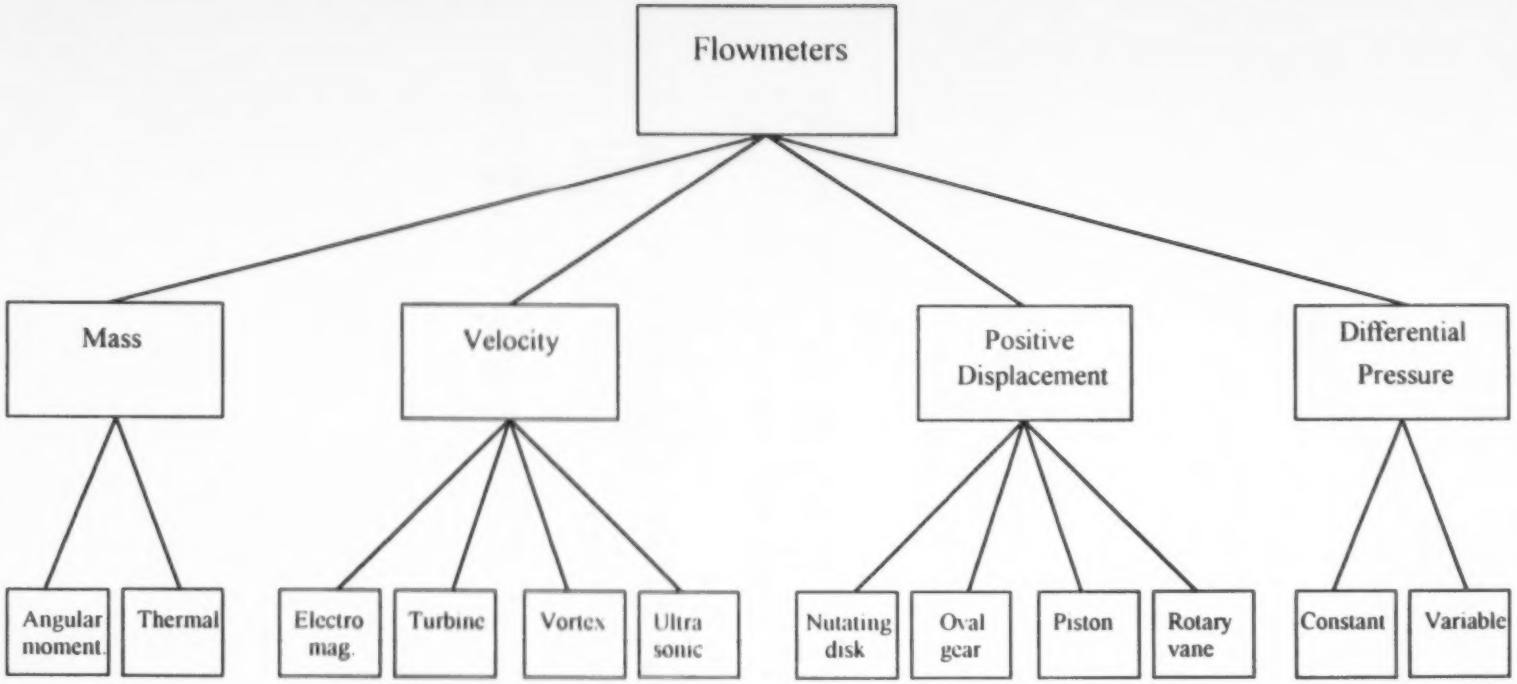

Figure 2.1 General classification scheme for flowmeters.

2.2.1 Overview

The mass and velocity meters are the latest techniques to become commercially viable thanks to the increased reliability and the decreased cost of electronic components. Mass meters differ from the other types of meters in that they do not require the density of fluid to calculate the mass flow rate. Angular momentum mass meters are based on Newton's second law of angular motion; they impart an angular momentum on the fluid and measure the resulting torque and angular velocity. Thermal mass meters are based on the heat transfer from a heated element positioned in the fluid (Lipták and Venczel, 1982).

Velocity meters measure flow by generating a signal that is linearly proportional to the volumetric flow rate (Meinhold, 1984). This can be represented as follows:

$$Q = \mathrm{K} \times \text{flowmeter dependent variable} \qquad (2.1)$$

where Q is the volumetric flow rate and K is a parameter dependent on the velocity flowmeter type and manufacturer.

Electromagnetic meters measure the voltage produced by a conductive fluid flowing through a magnetic field (Sovik, 1985). Turbine meters count the number of rotations made by a rotor whose axis is parallel to the direction of flow (Lipták and Venczel, 1982). Vortex meters measure the frequency of vortices induced into the flow (Meinhold, 1984). There are two types of ultrasonic meters: Doppler and transit-time. Doppler meters use the frequency difference between the transmitted wave and the received wave. The received wave is a part of the transmitted wave that reflects off entrained particles or a

second phase in the fluid. Transit-time meters determine the difference in time for a sonic pulse to traverse the fluid at an acute angle with the flow as opposed to the flow (Lipták and Venczel, 1982).

Positive displacement meters operate by splitting the flow into a number of discrete units of identical volume and counting the number of units that pass through the meter (Miller, 1983). Differences in meters relate to the mechanism used to break the flow into the individual units. These designs include nutating disk, oval gear, reciprocating piston and rotating vane (Meinhold, 1984).

2.2.2 Differential pressure flowmeters

Differential pressure meters are the most commonly used flowmeters in industry. In fact, according to a survey of the process industry conducted by Hasley (1986), they accounted for 78% of all flow metering devices. Three reasons for their extensive usage are maturity, versatility and simplicity (Ginesi, 1991).

Bernoulli presented the hydraulic equation in 1738 from which today's equations for differential pressure flowmeters have evolved (Miller, 1983). These equations are based on energy conservation with the key concept being that for a flowing fluid if the kinetic energy (energy associated with the velocity of fluid) is altered; then the potential energy (energy associated with the pressure of fluid) will change proportionally. In other words, when the velocity increases, the pressure decreases, and vice versa. In its simplest form, this can be expressed as

$$Q = kA\sqrt{\Delta P / \rho} \qquad (2.2)$$

or

$$\upsilon = k\sqrt{\Delta P / \rho} \qquad (2.3)$$

where k is a parameter dependent on the differential pressure flowmeter type; A is the area available for flow; ΔP is the differential pressure generated; ρ is the density of fluid; and υ is the average velocity of the fluid. A point worth noting is that the fluid density is required to calculate the velocity and volumetric flow rate (Lipták and Venczel, 1982). This is in contrast to mass and velocity type flowmeters neither of which require this parameter.

A differential pressure flowmeter is a general name that encompasses a wide variety of meters that are all based on the above principle. However in this thesis, only four types will be considered: variable area, V-Cone, flow nozzle and orifice plate. These types were selected because of their capability to handle a wide range of applications and operating conditions, and their current large usage in industry (the variable area and orifice plate types account for 19% and 56% of all flow measurements, respectively (Hasley, 1986)). In addition, since V-Cone, flow nozzle and orifice plate all have similar operating characteristics, there can be confusion as to which device is most appropriate for a given application.

These four differential pressure flowmeters can be further divided into two groups: constant differential pressure and variable differential pressure as shown in Figure 2.2. This division is based on the method used to measure the flow rate. Constant differential pressure flowmeters differ from variable differential pressure flowmeters in that they do not vary ΔP to calculate the flow rate. Instead, they maintain a constant ΔP and vary the area, A.

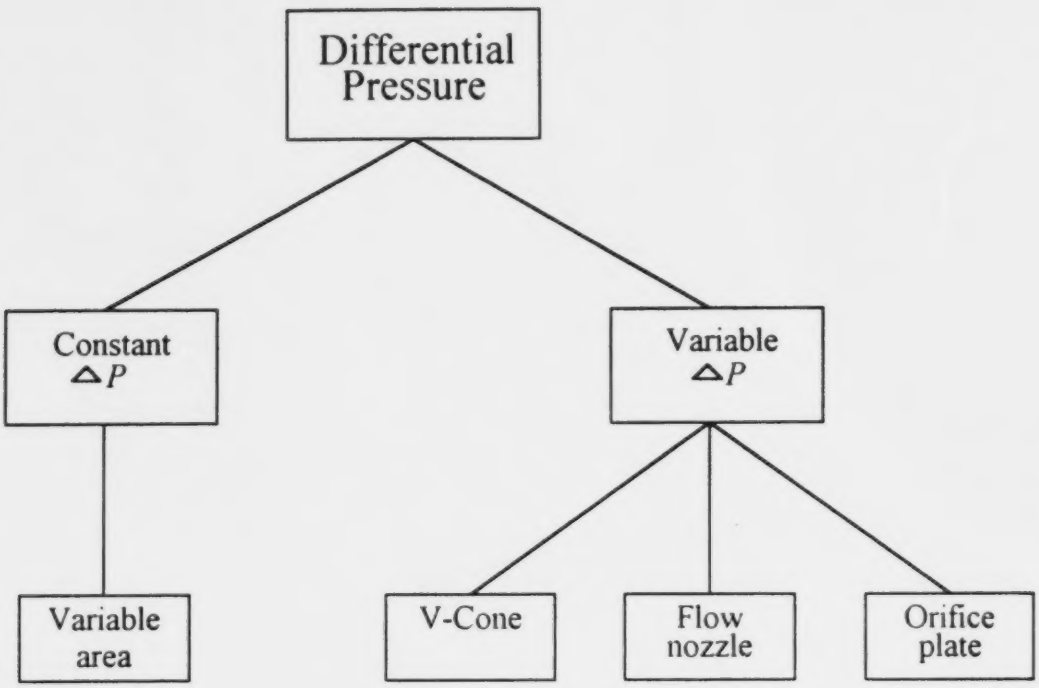

Figure 2.2 Classification scheme for differential pressure flowmeters.

One method† to vary the area is to place a float in a vertically mounted tube with a precisely cut slot down one side as displayed in Figure 2.3. When the meter is filled with fluid and there is no flow, the float rests on top of the tube completely obstructing flow through the slot. As the flow rate increases, the pressure drop across the float will increase until the upward acting forces (hydraulic and buoyancy) just balance the downward acting forces (weight of float). At this balancing flow rate and greater, the

---

† The most commonly known type of variable area flowmeter is the rotameter. However, this type was not selected for this study because no mathematical equations or computer programs were available for sizing them.

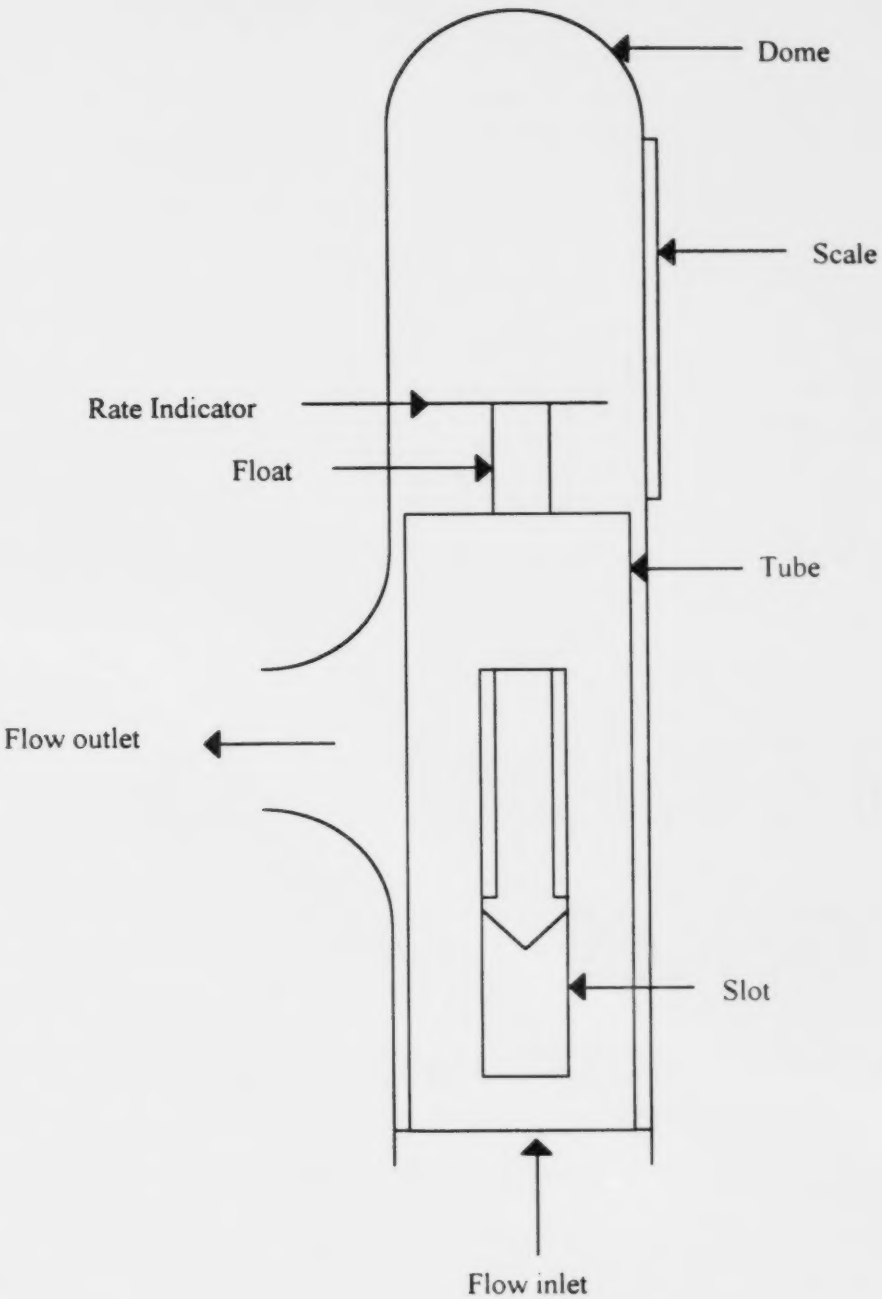

Figure 2.3 Schematic diagram of Meter Equipment Manufacturing, Inc variable area flowmeter.

pressure drop across the float will remain constant. However, at flow rates greater than the balancing flow rate, the float will rise, increasing the area for fluid to flow, until an equilibrium position is reached. At this position, the height of the float is linearly proportional to the flow rate of the fluid and can be used with a scale attached to the dome to provide a direct reading.

Conversely to constant differential devices, variable differential devices fix the area available for flow and allow the ΔP to vary. These types of meters consist of two elements: a secondary and a primary. The secondary element measures the differential pressure produced by the primary element. The primary element is an obstruction placed in the flow stream to decrease the area available for flow and thereby, generate the ΔP (Hayward, 1979). The V-Cone, flow nozzle and orifice plate differ in the shape of the obstruction used to generate the ΔP.

One of the new primary elements to become commercial available in the late 1980s is the V-Cone. As the name suggests, the V-Cone is shaped similar to a cone and is placed concentrically in the pipe with the tip of the cone pointed towards the on coming flow as shown in Figure 2.4. The V-Cone produces a pressure drop by reducing the area of flow to an annular space between the maximum diameter of the cone and the inner pipe wall. The differential pressure is measured between a pressure tap in the pipe wall just upstream of the tip and another in the centre of the downstream end of the cone.

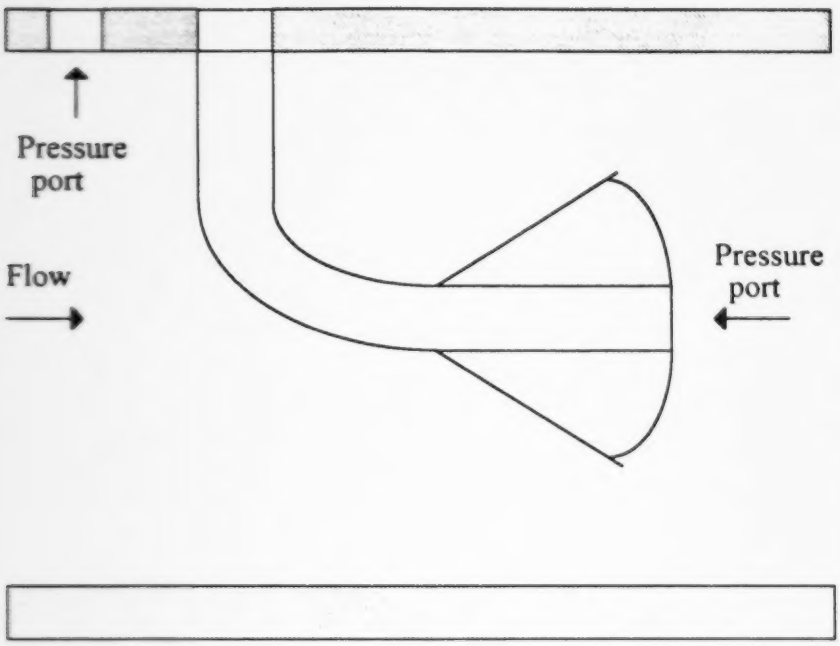

Figure 2.4 Schematic diagram of V-Cone flowmeter.

Flow nozzles produce a pressure drop by restricting the flow to a circular area concentric to the pipe. There are two distinct designs of flow nozzles, but only the type of design most commonly used in North America will be considered: the long radius or ASME flow nozzle. While both designs converge to cylindrical tube, the ASME flow nozzle shown in Figure 2.5 is characterized by a converging inlet shaped as a quarter ellipse. Generally, the differential pressure is measured between a pressure tap one pipe diameter upstream of the inlet face and another one-half pipe diameter downstream of the face (Lipták and Venczel, 1982).

Of all the primary elements, the oldest is the orifice plate. This element is also the simplest in design. Basically, an orifice plate is a thin circular plate with an opening to reduce the area of flow to less than the cross-sectional area of the pipe. The plate is inserted in the

pipe perpendicular to the direction of flow and the differential pressure measured by a pressure taps on either side of the plate. There are many types of orifice plates which vary in the location and shape of opening(s) and pressure tap locations. However, only the most commonly used type in North America will be considered namely the squared-edged concentric orifice plates with flange taps. The key characteristics of these plates are the concentric circular hole and the square edge on the upstream face as shown in Figure 2.6. The pressure taps, located one inch upstream and downstream of the plate faces, are in the flanges used to secure the plate in the pipe (Lipták and Venczel, 1982).

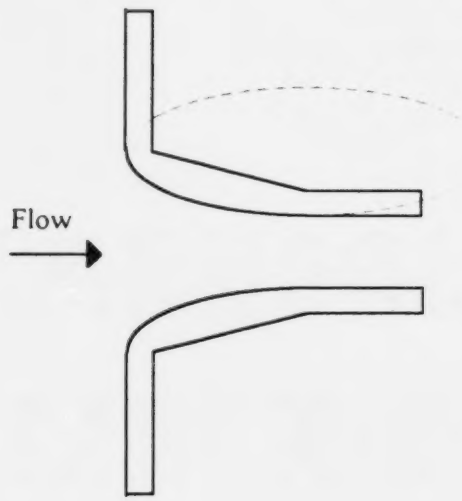

Figure 2.5 Schematic diagram of flow nozzle flowmeter.

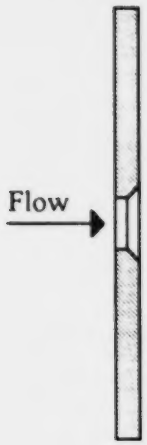

Figure 2.6 Schematic diagram of orifice plate.

2.3 Key Factors and Relationships

When selecting differential pressure flowmeters, there are many factors and relationships to be considered. It should be stated that the factors presented in this section are not all the factors to be considered but only some of the key ones. The factors can be categorized into three groups: system specifications, performance specifications, and cost specifications (O'Brien, 1989). Each of these groups will be discussed individually with regard to the factors that are included in that group and any relationships within the particular group.

Of the three specifications, the system specifications are the most stringent because they are determined by the type of process. In most cases, they cannot be altered. System specifications include fluid properties such as viscosity and density, fluid states such as liquid and/or gas; and process parameters such as temperature, pressure and composition. The fluid's properties and states are governed by the process parameters. In order to

predict these properties and states, various correlations have been developed relating the properties and states to process parameters.

The performance specifications employ terminology, which can be a source of problem (O'Brien, 1989; Rusnak, 1989), specific to the field of flow measurement. The terms include turndown, permanent pressure loss, repeatability and accuracy. Turndown is the ratio of the maximum to the minimum flow rate (e.g., 3 to 1). Permanent pressure loss is the decrease in static pressure as a result of friction as the fluid passes through the meter. Repeatability is the maximum amount of discrepancy expected between measurements of the same flow rate. Accuracy is the maximum amount of discrepancy expected when the flow rate measured by the flowmeter is compared with a very accurate measurement.

The cost specifications should consider not only the initial purchase price but also installation, operation and maintenance expenses. The most obvious cost is the purchase price. But the installation cost can be significant, and over the long term, cost of operation and maintenance may dominate. For example, the cost of the labour and equipment required to install an orifice plate in a small line (2 in.) or large line (60 in.) is greater than the plate itself. In the case of operational cost, the amount of energy required to compensate for the permanent pressure loss produced by the flowmeter can be considerable in large diameter pipes. And, if the loss of production because of a process having to be continually shutdown or shutdown for extended periods of time is included, the cost of maintenance can be significant.

An important equation is the Reynolds number. Over the years a range of Reynolds numbers has been determined for each type of variable differential pressure flowmeters over which each flowmeter type has proven to provide accurate measurements. The

Reynolds number, a dimensionless quantity, is the ratio of the flowing liquid's inertial forces to the drag forces. It is defined as

$$\mathcal{R} = \frac{D \times \upsilon \times \rho}{\mu} \qquad (2.4)$$

where $\mathcal{R}$ is the Reynolds number; D is the pipe diameter; υ is the average fluid velocity; ρ is the density of the fluid; and μ is the absolute viscosity of the fluid.

2.4 Problem Definitions

Although there are many designs of flowmeters, Lomas (1986) summarizes the current situation the best: "Each type of meter has its own specific advantages and limitations and no one meter combines all benefits and all features." In other words, no one meter is best suited for all applications. Therefore, each application must be evaluated individually.

The difficulties encountered in flowmeter selection and sizing can be broadly characterized as follows:

- large decision-making domain,
- numerous considerations, and
- ill-structured problem.

Furthermore, if a computer program is to be developed, other difficulties are

- inability to capture human expert knowledge, and

- inability to integrate different software packages.

As a result of the many diverse applications and large financial expenditures annually on flow measurement, it is not surprising that there are over 100 commercially available flowmeters (Miller, 1983) using a variety of techniques. This diversity of applications and multitude of meters create a large problem-solving space for meter selection which can be very time-consuming for a non-expert.

There are numerous considerations to be taken into account when selecting a flowmeter (Cheremisinoff, 1979; Lomas, 1977; Rusnak, 1989), some of which were presented in Section 2.3. The key to solving this problem is the determination of the pertinent set of factors for the specific application. When determining this set, an engineer should consider not only the expected normal operating conditions but also foreseeable changes in conditions that may affect the performance of the flowmeter (Spitzer, 1985). Also, there are overload conditions that may have to be accounted for at start-up and shutdown (O'Brien, 1989).

Although algorithmic type approaches have been proposed by Lomas (1977) and Meinhold (1984) to select flowmeter types for a specific application, neither approach has been widely accepted. However, a very general two-stage methodology has emerged (Lipták and Venczel, 1982):

1) select all the flowmeters that are suitable for the specific application, and
2) determine the optimal meter from those selected in stage 1.

This method is just a beginning, as the stages are too broad without a definite starting point, defined procedure, as well as definitive solution: all of which are characteristics of ill-structured problems (Corbin, 1992).

Flowmeter selection and sizing requires the handling of both numerical and symbolic information. The numerical information is often analyzed with numerical algorithmic programs. However, in order to process the symbolic information, an expert employs knowledge and symbolic reasoning. Because human knowledge and reasoning techniques are not well modelled by numerically based algorithmic programs (Rao and Qiu, 1993), it is difficult to capture them with conventional programming techniques.

Computer software packages are valuable sources of information when selecting flowmeters and provide a time-saving approach to sizing them. Therefore, it would be advantageous to combine these programs in a software environment. Unfortunately, these packages have been developed independently in different computer languages, run under different operating systems and hardware platforms. As a result, it is difficult to integrate these separate packages.

Chapter 3

Intelligent Systems

Artificial intelligence emphasizes processing non-numerical information and non-algorithmic methods (Buchanan, 1985), and is applicable to ill-structured problems (Rao, 1992). Although there are many areas in artificial intelligence, the one of interest in this thesis is intelligent systems. These systems can be classified into four groups: symbolic reasoning systems (expert systems), coupling systems, artificial neural networks and integrated distributed intelligent systems (IDIS) (Rao et al., 1993). In order to introduce the concepts used in this thesis, some background on expert systems, coupling systems and IDIS will be presented.

3.1 Expert Systems

Expert systems provide a programming technique which captures the knowledge and reasoning of human experts in a specialized field and applies them for non-experts. These systems utilize symbolic knowledge representation and processing, and consist of four separate but related parts: database, knowledge base, inference mechanism and user/developer interface as shown in Figure 3.1. This segregation facilitates independent design and modification of the components (Rao et al., 1989). In contrast, conventional programming techniques mostly use numerical data with mathematical operators and consist of two parts: data and algorithm. The control mechanism is an integral part of the

algorithm and cannot be altered independently. As an over simplified comparison between conventional programs and expert systems, conventional programming requires both information and the order in which this information is to be processed. Expert systems, however, require only the information not the sequence in which to process the information.

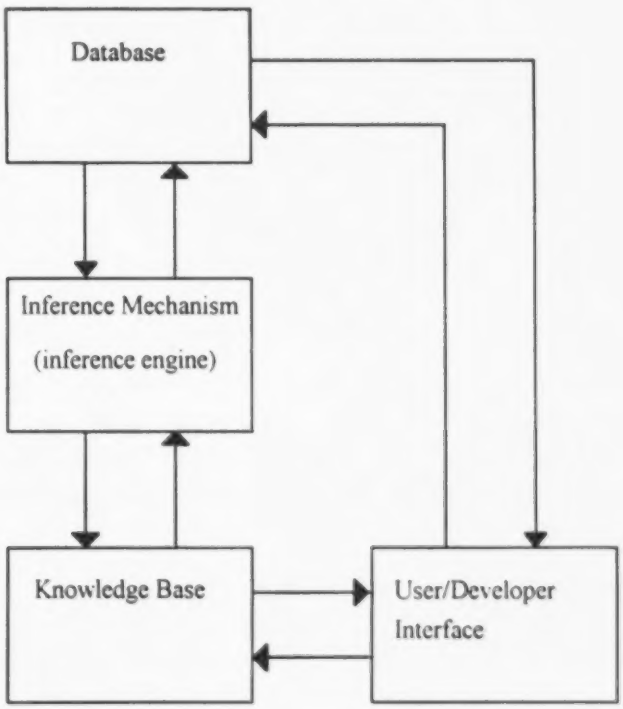

Figure 3.1 Components of an expert system.

The user/developer interface has one similar requirement for both user and developer: the interface must be easily intelligible. From the user's viewpoint, this means the interface must prompt for information and explain conclusions in a form the user can understand. From the developer's viewpoint, the interface must allow for easy development and editing of the knowledge base.

Symbolic knowledge representation refers to the mapping of a problem domain into a computer using symbolic constructs. Two prerequisites of the constructs are that they must be capable of capturing the domain knowledge and understandable by both the experts and non-experts. There are different forms of constructs, but most require selecting a symbol, a name or abbreviation that represents an object or variable, with variations on the means of assigning or associating a value with the symbol. These constructs are then formed into a knowledge base. The particular constructs utilized in this investigation will be described in Chapter 4.

Symbolic processing, also known as symbolic reasoning, is the operation of searching from an initial set of conditions through a knowledge base to obtain a solution. The choice of when to initiate and terminate the search and the path searched are all controlled by the inference engine. In order to determine a solution, the inference engine applies the value of symbols in the database (working memory) to the knowledge base (Rao and Qiu, 1993).

Initially[†], in order to implement expert systems, computer languages only familiar to the artificial intelligence community such as Lisp and Prolog were used. However, while these languages are efficient at symbolic reasoning, they lack the numerical capabilities of more conventional languages such as C, BASIC and FORTRAN. This lack of numerical computational power is a major drawback because solving complex engineering problems

---

[†] Currently, the trend is to use more conventional programming language such as Pascal and C which are able to create records and structures, respectively.

requires both numerical and non-numerical information (Kitzmiller and Kowalik, 1987) as depicted in Figure 3.2 (Rao et al., 1993). As a result, coupling systems were developed.

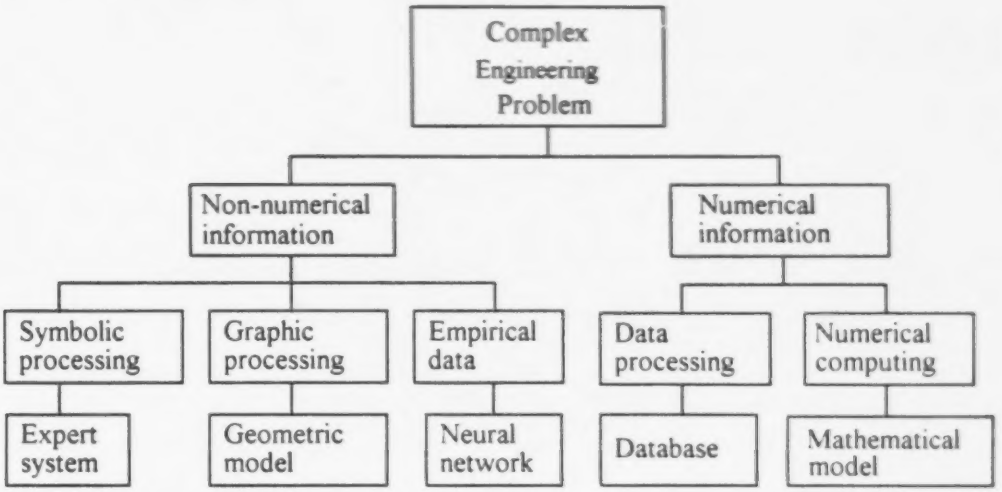

Figure 3.2 Non-numerical and numerical information to solve a complex problem.

3.2 Coupling Systems

Coupling systems can generally be defined as programs that combine symbolic processing and numerical computing as illustrated in Figure 3.3 (Wong et al., 1988). Although this is a very broad definition and not agreed on by all (Kitzmiller and Kowalik, 1987), it provides insight into what coupling systems are. The key feature is that two different but complementary techniques are brought to bear on a problem: the symbolic techniques of expert systems and the numerical capabilities of conventional programs.

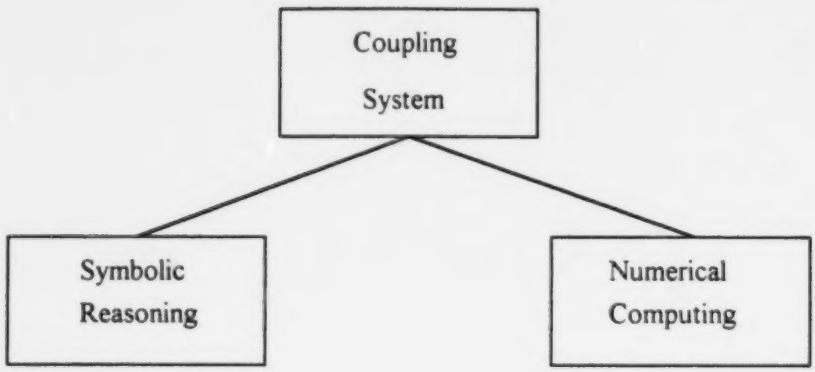

Figure 3.3 Illustration of a coupling system.

There are many applications for coupling systems. One application is to reduce the time required to solve a problem (Wong et al., 1988). The most obvious approach is to let the expert system perform only symbolic reasoning and use a conventional language to perform numerical computations. An alternative approach is to allow the expert system to reduce the search space (Rao et al., 1988) before performing numerical calculations.

Another application is to use symbolic techniques to integrate a number of independent numerical programs in order to solve a problem. These types of systems have two objectives. One objective is to utilize the symbolic system to manage the individual numerical packages so as to solve a problem that none of the packages can solve alone. Another objective is to utilize the symbolic system as an intelligent interface which assists users in selecting the appropriate numerical package for the specific task (Kitzmiller and Kowalik, 1987).

3.3 Integrated Distributed Intelligent Systems

The concept of an IDIS was proposed by Rao et al. (1987). Over the past several years, it has been redefined until it can now be stated as follows (Rao et al., 1993):

> Integrated Distributed Intelligent System is a large-scale knowledge integration environment, which consists of several symbolic reasoning systems, numerical computation packages, neural networks, database management subsystem, computer graphics programs, an intelligent multimedia interface and a meta-system. The integrated software environment allows the running of programs written in different languages and the communication among the programs as well as the exchange of data between programs and database. These isolated intelligent systems, numerical packages and models are under the control of a supervising intelligent system, namely, the meta-system. The meta-system manages the selection, coordination, operation and communication of these programs.

Figure 3.4 illustrates this architecture.

IDIS provides many desirable features:

- communication between systems,
- multi-media interface for displaying information,
- relative ease of maintenance and additions to the system,
- efficient knowledge representation and organization, and
- integration of existing software packages (Rao et al., 1993).

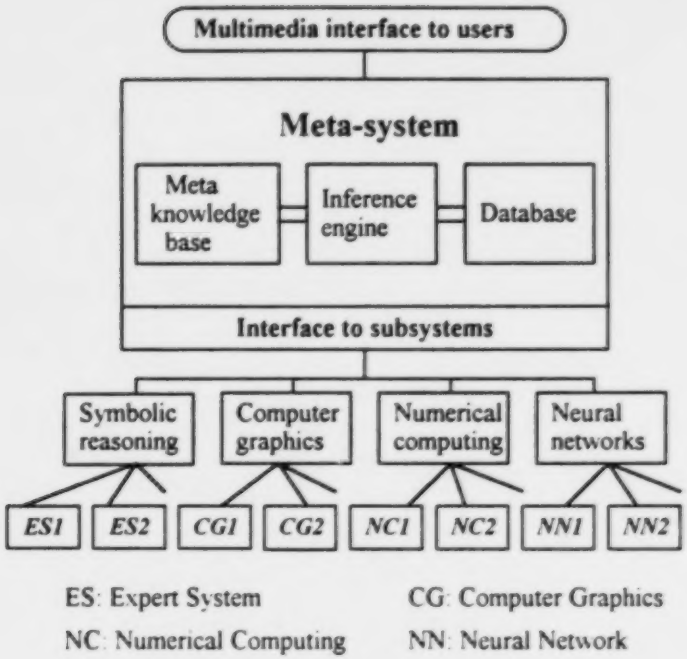

Figure 3.4 IDIS architecture.

The key component of an IDIS is the meta-system. It provides four functions that are applicable in the current study: coordinator, communicator, distributor, and integrator. As a coordinator, the meta-system manages each of the subsystems by controlling the selection and the initialization of each sub-system as required. As a communicator, the meta-system interprets and translates information (whether it is supplied by the meta-system itself, another sub-system or a user) into a format required by a particular sub-system. This may involve converting numerical data into symbolic information or passing numerical data in a pre-determined format. As a distributor, the meta-system facilitates maintenance by separating the sub-systems into independent systems that can be developed and debugged individually. Also, this distribution can reduce the solution time as only knowledge pertinent to the current problem is applied. As an integrator, the meta-system provides an approach whereby new knowledge and subsystems can be easily

added. Since the meta-system controls the ordering and interfacing between subsystems, new subsystems can be added without altering existing sub-systems (Rao et al., 1993).

When one looks at Figure 3.4, the meta-system appears similar to an expert or coupling system in that it has a database, knowledge base and an inference engine. However, in order to provide all the features and functions, these three components are much more complex than in an ordinary expert system or coupling system. This complexity will become apparent in the following chapter where the implementation of the meta-system is discussed.

Chapter 4

Implementation of a meta-system, Meta-COOP

In order to implement a meta-system, Meta-COOP has been developed (Rao et al., 1993). Meta-COOP is coded in C and based on OOP techniques. Meta-COOP is a hybrid environment that incorporates frames and production systems to capture human experts' knowledge and perform symbolic reasoning.

This chapter begins with a brief overview of the Meta-COOP architecture in order to illustrate the differences between Meta-COOP and a standard expert system shell or coupling system. This is followed by an introduction to the basic concepts of OOP. The chapter describes the different knowledge organization and representation schemes in Meta-COOP.

4.1 Meta-COOP Architecture

Although Meta-COOP operates as an integrated unit, its architecture as displayed in Figure 4.1 (Rao et al., 1993) can be categorized into four groups: knowledge representations, databases, interfaces and inference engines. Meta-COOP has three types of databases: database, local and global fact bases. The database in the Meta-knowledge base is analogous to a database file in conventional programming where static data (e.g. records) are stored permanently. The global and local fact bases are similar to a database

(working memory) in an expert system in which they both act as temporary storage for intermediate information produced by dynamic processes. The global fact base posts results, determined by the different subsystems, which are accessible by all inference engines. The local databases store facts that are only accessible by the subsystems that create them, thereby one of the concepts of OOP, i.e. encapsulation is implemented.

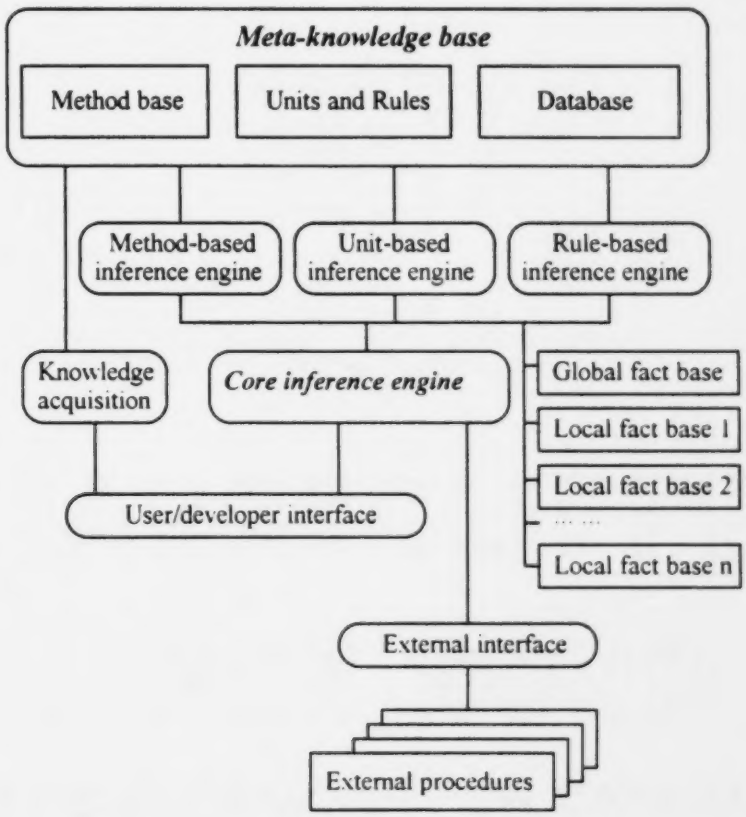

Figure 4.1 Meta-COOP architecture.

In Meta-COOP, the different interfaces enable the transfer of information to and from the external environment. The user/developer multimedia interface provides two functions: (1) a means for a user to input information and for the system to output results during a

consultation and, (2) allows a developer to create and edit the meta-system knowledge base. The external interface provides communication to independent systems. These independent systems are external procedures coded in C or an executable file.

In contrast to an expert system shell or coupling systems, Meta-COOP consists of four types of inference engines. The core inference engine is responsible for sending internal messages, initiating the other inference engines and invoking external procedures through the external interface. The other three inference engines, rule-based, method-based and unit-based, operate only when the particular type of knowledge representation for which each was developed is being processed. The rule-based engine, i.e., the inference engine in most successful expert systems or coupling systems (Rao et al., 1988), performs inferences on symbolic knowledge contained in rule slots. The method-based engine operates on algorithmic type programs contained in method slots. The unit-based engine provides inferences using knowledge contained in the hierarchy structure.

4.2 Object-Oriented Programming (OOP)

Over the past several years, OOP has become very popular due to the following reasons: increased programmer productivity, and easier maintenance and extendibility of programs (Rutz, 1991). The increased productivity can be attributed to the fact that programmers can create their own data types (Wiener and Sincovec, 1984) and are not limited to only data types such as strings, integers and real numbers provided in conventional programming languages. The easier maintenance of programs comes from the fact that programmers can make changes to a class of objects without affecting other classes, thereby altering a number of objects at once without concern for communications. The easier extendibility is achieved since the programmer can add new objects without

affecting existing objects. In addition, the programmer can create more specific objects by making only minor modifications to an existing class (Rutz, 1991). In order to understand how to accomplish the above functions, the features of OOP are presented in the following material.

4.2.1 Objects and classes

Objects are defined as entities in the real world that have been mapped into computer software. For example, objects can represent common geometric shapes such as circles and squares or, more relevant to the present work, flowmeters. However, to classify as objects, they must be capable of manipulating the data within an object. This manipulation of data is performed by method(s), there can be more than one method per object, which are also contained within the object (Rutz, 1991, Stefik and Bobrow, 1986). Thus, objects combine data and methods.

The object's data can be string (alpha-numeric) or numerical (integer, real) and either variable or constant. The methods are the algorithmic subroutines or functions that use the data in the object. In OOP terminology, the method is defined to manipulate the object. The methods contain instructions in the form of mathematical expressions, assignment statements and control commands similar to those in conventional programming languages such as FORTRAN and BASIC. As an example, a geometric object can contain methods for displaying its shape on a screen, printing its outline on a printer and storing its image in memory. As another example, more pertinent to this thesis, as an object, a flowmeter can contain a method for calculating the Reynolds number.

In OOP, the objects that are spawned from a *class* are called instances of the *class*. The class groups objects (instances) that have similar attributes and methods (Rutz, 1991). As Ten Dyke and Kunz (1989) so concisely stated: "A *class* is a template from which objects [instances] are created." A class determines the data and methods that all instances of the class will possess. However, objects can differ in the values assigned to the variables. As an example, using Figure 4.2, the class, orifice-plate, has spawned two instances: orifice-plate-1 and orifice-plate-2. Both instances contain the same data, pipe_diameter and bore_size, and method, flow rate as defined in the class. However, in the instance orifice-plate-1, the data pipe_diameter and bore_size have the values 6 and 3, respectively; while in the instance orifice-plate-2, the data pipe_diameter and bore_size have the values 2 and 0.5, respectively.

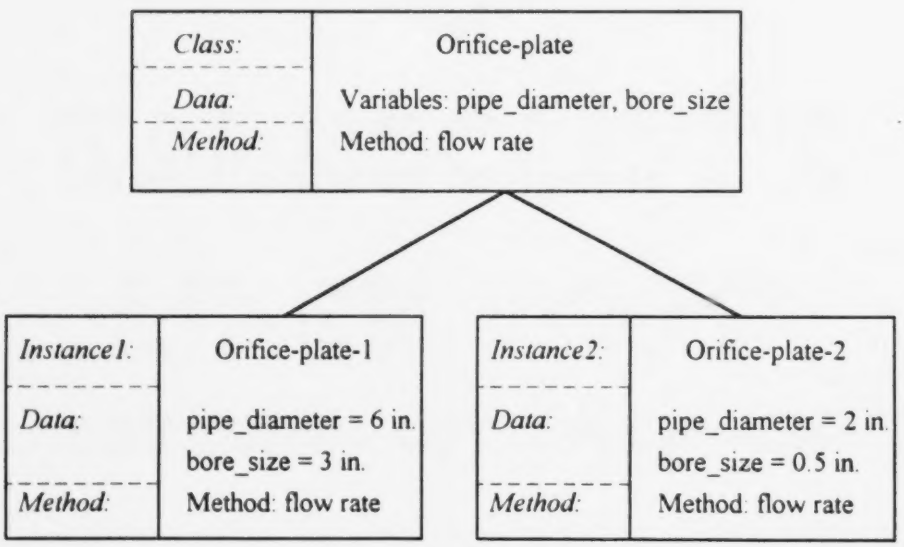

Figure 4.2 Example of a class and instances.

4.2.2 Hierarchical inheritance

Hierarchical inheritance is an important feature that allows a more specific class, known as a subclass, to be derived from a more general class, known as a superclass (Rutz, 1991; Ten Dyke and Kunz, 1989). In deriving a subclass, only new data or methods unique to the subclass need to be defined since all data and methods of its superclass automatically become associated with the subclass†. This process of a subclass inheriting data and methods from its superclass can pass through many levels. Figure 4.3 illustrates this process. The variable "pipe_diameter" is declared in the superclass "differential-pressure-flowmeters"; the variable ΔP is declared in the class "variable-differential-pressure-flowmeters"; and the variables bore_size and flow_area are declared in the subclass "orifice-plate" and "v-cone", respectively. However, because of hierarchical inheritance, the instances "orifice-plate-1" and "v-cone-1" both contain the variables pipe_diameter and ΔP. In addition, these instances contain the variables unique to the class that spawned them. Thus, "orifice-plate-1", an instance of "orifice-plate", contains the variable "bore_size" and v-cone-1, an instance of "v-cone", contains the variable "flow_area".

4.2.3 Encapsulation, message sending and data abstraction

Encapsulation or information hiding refers to the private nature of an object's data. In OOP languages, the object's data can only be accessed or altered by the methods attached to the object. In other words, the value of a variable cannot be modified without

---

† A subclass differs from an instance in that the subclass contains additional data and methods, and not just different values assigned to the variables.

employing the methods associated with the object. Therefore, the only way to alter a variable is to send a message to the object.

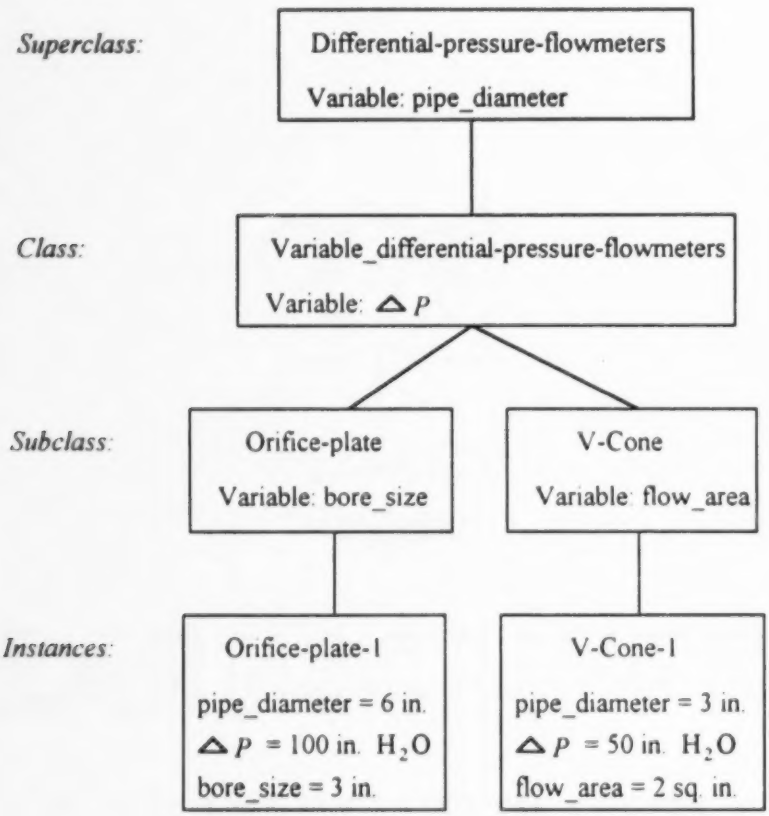

Figure 4.3 Example of superclass-subclass inheritance.

As Stefik and Bobrow (1986) stated, the key to "All of the action in object-oriented programming comes from sending messages between objects." For an object-oriented program to proceed one object must send a message and another object must receive the message. Messages consist of at least two parts†: the name of the receiving object and

† Some OOP allow the passing of arguments in the message.

the name of the method. The receiving object processes the message by performing the instructions in the named method.

The use of message sending to communication between objects leads to another fundamental concept of OOP: data abstraction. Data abstraction refers to the fact that an object sending a message needs the name of an object and method, but it requires no knowledge of the data structure of the receiving object. Thus, the receiving object is viewed as a black box from the sending object's perspective.

In summary, the three concepts that have been discussed are advantageous to programmers because they are unconcerned about an object's data being altered by another part of the program or the data structure of the object; they only need to know how the object responds to a particular message.

4.3 Knowledge Organization and Representation†

When solving complex engineering problems, problems are often too large and complex to understand as a whole. For this reason, we tend to decompose (divide) such problems into parts that can be easily understood and establish communications between the parts so that the overall problem can be handled. In order to accomplish this, two organizational schemes are utilized: decomposition (partition) and classification (Rao et al., 1993). Decomposition refers to division of the problem into smaller subproblems that are easier

---

† This section is intended to illustrate the generic structure and form of knowledge organization and representation in Meta-COOP. For the specific implementation, the reader is referred to Appendix A which contains the knowledge base for IDISDPFSS.

to solve than the original problem. Each subproblem solution is an integral part of the overall solution, but is only weakly related to other subproblems. Classification refers to the grouping of components to solve a subproblem. These components are closely related to each other. As an over simplified example, if one considers the design of an airplane and support facilities as the problem, it can be decomposed to design of the airplane and design of the runway. The design of the airplane is weakly related to the runway in that the runway must be long enough for the airplane to takeoff and land. However, the components that contribute to the design of the airplane are much more closely related and would be grouped together.

In essence, decomposition provides a partitioning of the upper most levels and a hierarchical knowledge representation. When moving horizontally in the hierarchy, functional differences are evident. When moving vertically down the hierarchy, increasing detail is exhibited.

As illustrated in Figure 4.4, Meta-COOP accommodates decomposition by allowing a problem (project) to be divided into knowledge bases. Each knowledge base contains part of the solution to a problem. A knowledge base is comprised of one or more units that further decompose the problem. These units are the structures that Meta-COOP provides for classification.

A unit† is the primary element used to organize and represent knowledge in Meta-COOP. The unit is analogous to an object in OOP in that both support the combining of data and methods, instances, encapsulation, inheritance and data abstraction. However, a unit differs from a class in that data representations and processing are more advanced and aimed at capturing knowledge and reasoning abilities of experts.

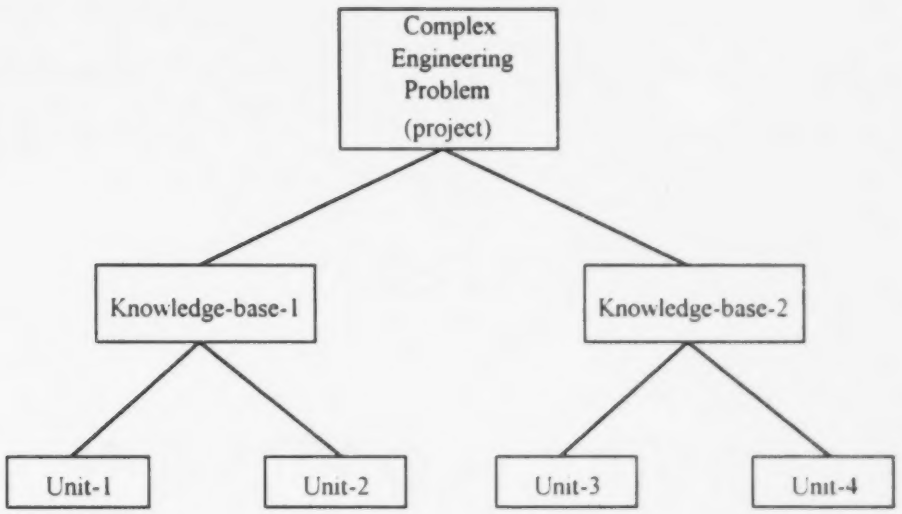

Figure 4.4 Decomposition of a complex engineering problem.

Figure 4.5 illustrates the header of a unit. Each unit's header must have a declaration section, Globe - End, and title statement. The declaration creates members and variables that are in existence for the entire duration of the program and makes them accessible by all sections of the program. These items are stored in the global fact base. The members

---

† A unit in Meta-COOP is similiar in form to a frame in other systems, KEE™ (Fikes and Kehler, 1985). However, since when declaring a frame in Meta-COOP, the word unit is stated, the term unit will be used to describe this knowledge structure in this thesis.

are instances of the unit and analogous to instances in OOP. The variables are similar to variables in FORTRAN and BASIC and must include the data type (integer, real or string). The title (UNIT) states the unit-name and the knowledge base to which it belongs. Optionally, the header may contain a component subclass that establishes an inheritance relationship between units. This inheritance relationship forms a superclass-subclass hierarchical structure analogous to inheritance in OOP. Thus, subclass (descendent) units are able to inherit the slots from a superclass unit.

```
GLOBE

member-name: unit-name;
variable-name: integer, real or string;

END

UNIT: unit-name      in_knowledge_base    knowledge-base-name;

Subclass:      descendent-unit-name;

Slots

End Unit;
```

Figure 4.5 Structure of a unit's header.

Units are composed of different types of slots. Each slot must have a title that identifies the slot name and the unit it is associated with. Slots also possess facets. Facets identify the type of slot and contain data (attributes), knowledge (rules) and procedure (methods).

Figure 4.6 illustrates one type of slot: an attribute. An attribute describes a characteristic of the unit. An attribute is similar to a constant or variable in OOP where it is assigned a value. An attribute differs from a constant or variable since it has three facets: valueclass,

inheritance and value. The valueclass records the data type (integer, real or string). The inheritance assists in distinguishing the attribute slot from other slot types. The value is the data.

```
MEMBERSLOT: Max_press from    user_input;
        Inheritance: OVERRIDE.VALUES;
        Valueclass: real;
        Value: Unknown;
End Slot;
```

Figure 4.6 Example of an attribute

As illustrated in Figure 4.7, a second type of slot, similar in structure to an attribute slot, contains the name of an instance of a unit in the facet's valueclass. This type of slot is employed to graphically display the decomposition structure of a problem to a user and will be illustrated in Chapter 7.

```
MEMBERSLOT: Max_press from    user_input;
        Inheritance: OVERRIDE.VALUES;
        Valueclass: initial_check;
        Value: Unknown;
End Slot;
```

Figure 4.7 Example of a decomposition slot

Figure 4.8 illustrates another type of slot: methods. More than one method can be attached to a single slot. To specify the slot as a method slot, the facets inheritance and valueclass are set to METHOD and METHODS, respectively. The methods themselves

consist of three parts: header, declaration and body. The header section contains the method name, "choose_method", and a keyword, "choose", used in the message sending. The declaration section is where the local variables (e.g. "x") and their types (integer, real and string) are specified and stored in the local fact base. In contrast to global variables, local variables only exist during execution of the method and expire on termination of the method. The body is implemented in a language that is a subset of the computer language Pascal and contains two additional statements for message sending and invoking symbolic reasoning. The syntax of the message sending statement is as follows:

```
send ("check") to "initial_check";
```

where "check" is a keyword in a method contained in an instance of a unit named "initial_check". The syntax of a statement to invoke symbolic reasoning is described as follows:

```
reason(_FRAME,"check_rules");
```

where _FRAME is a variable that is replaced by an instance of the unit; and check_rules is the memberslot that contains the rules.

The methods in a unit perform the same function as the methods in OOP. They are algorithmic type programs that contain mathematical expressions, control commands and message sending. However, methods in a unit differ from those in OOP in that unit methods can invoke rule slots to perform symbolic reasoning.

```
MEMBERSLOT: choose_method    from    flowmeter;
        Inheritance: METHOD;
        Valueclass: METHODS;
        Value: choose_method;
End Slot;


METHOD choose_method (choose:keyword)


VAR
x: integer;
BEGIN

 send ("input flow_data") to "req";
 _FRAME.initial_check:="initial_check";
 send ("check") to "initial_check";
 if initial_check.check<>"fail" then
 begin
   ...
 end;
END.
```

Figure 4.8 Example of a method slot.

Illustrated in Figure 4.9 is a rule slot. A rule slot presents expert knowledge and reasoning in the form of a production system. A production system stores knowledge symbolically in rules. Each rule has two elements: a premise (fact) and conclusion (then). The premise is the conditional or test element and can contain logic operators ("and" and "or"), inequalities (>, <, >=, <=, <>) and equality (=). If the premise is true, the rule is said to fire and the conclusion is executed. The conclusion is one or more assignment statements and can contain arithmetic operators (+, −, ×, ÷). The symbolic reasoning proceeds by inference through the rules.

```
MEMBERSLOT: check_rules          from    initial_meters_check;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: RULES;
  Value: {
        rule 1
          fact req.Max_press>=6000.0
          then _FRAME.check="fail"
                _FRAME.check_default="fail"
        rule 2
          fact req.Max_temp>=93.33
               req.Max_press>=5830.0
          then _FRAME.check="fail"
                _FRAME.check_default="fail"
        ...
End Slot;
```

Figure 4.9 Example of a rule slot.

Chapter 5

Individual Software Packages

The first four chapters in this thesis defined the problems in differential pressure flowmeter selection and sizing, discussed expert systems types and the concept of integrated distributed intelligent systems, as well as presented the Meta-COOP environment. In the following three chapters, the implementation techniques for IDISDPFSS (integrated distributed intelligent system for differential pressure flowmeter selection and sizing) will be discussed. This chapter presents the individual software packages that have been integrated. Chapter 6 describes the communication between heterogeneous computer hardware platforms and operating systems. Chapter 7 discusses the IDISDPFSS interface and provides an illustrated demonstration.

The concept of IDIS involves the integration of software packages that have already been developed for specific purposes. Such an integrated environment is controlled by a meta-system. This integrated environment can solve more complex problems that are difficult to be solved by the individual packages separately. Before discussing IDISDPFSS, the following terminology in this chapter is defined:

In-house developed software: This is software developed by the author for a specific purpose using a high-level programming language e.g. C or FORTRAN.

Commercial software package: This software is developed by a commercial company for a specific purpose and distributed to users. The users often obtain an executable file and formulated input/output files.

AI tool: This is an expert system development environment for building an expert system for specific purposes.

Coupling system: This is an independent expert system developed using an AI tool that combines symbolic processing and numerical computations.

IDISDPFSS integrates in-house software, two commercial software packages, and a coupling system. The in-house software, MEM 1.0, for sizing MEM (Meter Equipment Manufacturing, Inc.) variable area flowmeters was developed by the author. Two commercial software packages, V-Cone 3.1 and FLOWEL® 2.0, were used to size the V-Cone flowmeters, and the flow nozzles and orifice plates, respectively. The coupling system can implement selection of viscosity models and prediction. In this chapter, the capabilities and key features of these individual software packages that are integrated in IDISDPFSS are presented.

5.1 Variable Area Flowmeters

In order to size the variable area flowmeters, an autonomous computer program (MEM 1.0) has been developed based on mathematical equations supplied by the Meter Equipment Manufacturing, Inc. (MEM). These equations convert a flow rate at operating conditions to a flow rate at MEM base conditions. Once the flow rate at MEM base conditions is calculated, it is used to determine the proper size of flowmeter.

There are three equations, each for different types of applications, the choice of which depends on the state and composition of the fluid. If the fluid is in a liquid state, equation 5.1 is applicable for all compositions. If the fluid is in the gas state and not steam, equation 5.2 is selected. In the case of steam, equation 5.3 is applied.

$$Qm = Qa\sqrt{\frac{da(df - dm)}{dm(df - da)}} \qquad (5.1)$$

where Qm is the flow rate (U.S. GPM) at MEM base; Qa is the flow rate (U.S. GPM) of the liquid to be metered at operating conditions; da is the density (lb./ft.^{3}) of the liquid being metered at operating conditions; df is the density (lb./ft.^{3}) of the flowmeter float; and dm is the density (lb./ft.^{3}) of the liquid at 70°F and in standard atmosphere.

$$Qm_g = Q_g \times 0.465 \times \sqrt{\frac{SG_g \times T_g}{P_g}} \qquad (5.2)$$

where Qm_g is flow rate at 70°F and 100 psig; Q_g is the flow rate of gas to be metered at operating conditions (SCFM); SG_g is the specific gravity of gas (relative to air) to be metered at 70°F and 14.696 psia; T_g is metering temperature (Rankine); and P_g is metering pressure (psia).

$$Q_s = Qm_s \times \sqrt{Sv} \times 0.17 \qquad (5.3)$$

where Q_s is the required MEM meter capacity (SCFM), Qm_s is the flow rate (lb./hr.) of steam; and Sv is the specific volume (ft.^{3}/lb.).

In order to query a user for information, calculate the MEM flow rate based on the above equations, and display results, a computer program with a multi-window user interface has been developed. The program utilizes C-WIN (1993) version 1.0, a public domain software package written in C that contains a number of routines to create and manipulate windows on IBM® personal computers and compatibles running PC-DOS® or MS-DOS ®†. While C-WIN provided routines to develop a windowing interface, it did not possess the capability to identify the user's keyboard input and control cursor movement. Therefore, it was necessary to add this functionality. A copy of the source code for the main program is presented in Appendix B.

Although there are other packages for developing window applications, C-WIN is chosen for five reasons. One reason is the cost since it is free. Another reason is that the package does not make any calls to special graphic routines. This is important because of the communication set up between Meta-COOP and the program, which will be discussed in Section 6.1.1. The third reason is that the package could run on many different models of personal computers and display screens without difficulty. The fourth reason is that since the source code was available, it was possible to add the required keyboard functionality. The fifth reason is that there is no copyright protection on the package. Reasons three through five are important because the sizing program, complete with source code, is to be supplied to MEM for possible further modification and distribution for a PC hardware platform and DOS operating system.

---

† For the remainder of this thesis, the Disk Operating System (DOS) of either Personal Computer (PC) or Microsoft (MS) will be simply referred to as DOS.

5.2 V-Cone Flowmeters

For sizing the V-Cone flowmeters, Ketema/McCrometer Division supplied a commercial software package, V-Cone version 3.1. Since this software package contains proprietary information, only the executable code is available and is to be used without modification. This package was written in BASIC for IBM personal computers and compatibles running DOS. V-Cone 3.1 queries the user for input and based on this information calculates one of the following: inside diameter of the pipe, flow rate, size of cone or generated differential pressure.

The user interface is a single display screen with multi-sections, each displaying specific information. One section enables the user to specify requirements, to display them in the other sections, and to initiate commands. This section has five components. The ID & Order accepts the meter identification name, tag number, customer name and computer file name. The "Program File Manager" presents commands for printing results, loading and saving files, clearing data and initiating calculations. The "Rate & Diff" Pressure allows values to be entered for these two parameters. The "Meter Specifications" enables the style of meter, pipe and cone size, and flow turndown to be specified. The "Fluid Specification" is where fluid properties such as temperature, pressure, density and viscosity are entered.

In order to assist the user with the determination of the fluid's density and viscosity, the program has the capability to estimate these values for some fluids. For water, steam and air, the program can estimate both the density and viscosity at the specified temperature and pressure. For liquids other than water, if its molar mass and critical temperature and pressure are known, the density can be estimated but not the viscosity. For a gas, if it is

one of the 21 gases (e.g. CL_2, He, NH_3 or CO_2) in the database or if its molar mass, critical temperature and pressure are known, both the density and viscosity can be predicted.

5.3 Flow Nozzle and Orifice Plate Flowmeters

The ASME flow nozzles and concentric orifice plates with flange taps are sized using the same package: FLOWEL 2.0 a commercial package donated by Kenonic Controls Ltd. This package also has the capability to size concentric orifice plates with pipe, corner or radius taps; quadrant edged, segmental, eccentric and restriction orifice plates; classical venturi tubes with rough-cast, machined or rough welded sheet-iron convergent; ISA 1932 flow nozzles; ISO venturi nozzles (Wilcox, 1988). However, because the IDISDPFSS only considers two types of flowmeters contained in FLOWEL 2.0 and the extensive features of the program, only a very brief description[†] will be given including features relevant to this thesis.

FLOWEL 2.0 has a multi-window user interface which prompts the user to supply information. The information is contained in two main sections: "The Main Program Screens" and "The Setting Screens". The main program screens are a series of six sequential screens that collect and display requirements, such as temperature, pressure and viscosity and the calculated results. The results can be calculated using three different methods: American Gas Association (AGA) 3, International Organization for Standardization (ISO) 5167 and General Application. The choice of method is based on the application and user preference. The setting screens, of which there are eight, differ

---

[†] For a more detailed description the reader is refered to Wilcox (1988).

from the main program screens in that the user need not enter these screens for some applications, and information contained in these screens is not displayed to the user prior to performing calculations. Four of the setting screens are used to input data for density and compressibility calculations (Wilcox, 1988).

The density of a fluid can be entered manually or calculated using one of eight methods. The selection of which method to employ is based on the state of the fluid, information available and user preference. For liquids, only one method is applicable and requires the specific gravity of the liquid. For gases, three methods (ideal specific gravity, molar mass [molecular weight in the program] and real specific gravity), all based on the ideal gas law, are available. All three methods require the temperature, pressure and compressibility factor and either a specific gravity or molar mass. There are also four other more sophisticated procedures to calculate density and compressibility factor. Besides temperature and pressure, these methods require the fluid's composition. Two of the methods, Wichert-Aziz and Redlich-Kwong, require the complete fluid composition while two others, AGA 8 and NX-19, can use the complete composition or only key component compositions with specific gravity and/or heating value (Wilcox, 1988).

5.4 A Coupling System for Selection of Viscosity Models and Prediction

The viscosity is an important fluid property used in many engineering applications that involve fluid movement (Peng and Vermani, 1987). In flowmeter selection, it is required to calculate the Reynolds number which is one of the key factors in determining the applicability of variable differential pressure flowmeters such as V-Cones, flow nozzles and orifice plates. However, accurate determination of a fluid's viscosity may not be an easy task. Therefore, a coupling system has been developed that can not only select an

appropriate viscosity model for a specific fluid but also predict the viscosity of the fluid based on the selected model.

Although a fluid's viscosity can be determined from literature sources or experimental results, the primary source is model-based methods. Literature sources are limited in the fact that the values for a complex fluid multi-component mixture at a specific operating temperature and pressure may not be available. Experimental results suffer from the fact that they are time-consuming and expensive to perform. Model-based methods are preferred because they are suited for computer implementation using conventional programming languages and can predict values for simple and complex fluids over a wide range of temperatures, pressures and compositions.

Over the years, based on different theories, many investigators have developed a number of viscosity models. However, no one model is capable of reliably predicting the viscosity of all the diverse fluids encountered in engineering applications (Peng and Vermani, 1987). The various models, in general, can be grouped into two major categories: (i) empirical correlation and (ii) corresponding state principle. Basically, the models, such as Jossi et al. (1962), Dean and Stiel (1965), Tham and Gubblins (1970), etc., developed by empirical correlation are limited to narrow ranges of temperature and pressure, and often to a simple pure fluid. Thus, they cannot be applied to such practical cases as a petroleum liquid, polymer solution, etc. The other models, such as Ely and Hanley (1981), Pedersen and Fredenslund (1987), etc., are based on the corresponding state principle in which the properties of the fluid of interest are evaluated with respect to a given simple reference fluid. These models are reliable in estimating the viscosity of complex mixtures/fluids as well as simple fluids. However, it is not worth using these methods to predict the viscosity of a pure, simple fluid for the following reasons: (i) Usually, the performance is time-

consuming due to the complexities of these models. (ii) The models based on empirical correlation can also produce reliable results for those simple fluids while repetitious calculations are usually not encountered with these methods (P. Du, personal communication, Mar. 1993).

Since the task of viscosity prediction is specialized, most engineers are not knowledgeable in all the viscosity prediction methods, their limitations and the varying computational time requirements. While these non-experts may choose a method based on familiarity or availability, experts base their selection more on the specific application (Gani and O'Connell, 1989).

5.4.1 Numerical computations

This coupling system can select one of three viscosity models, Dean and Stiel, Ely and Hanley, Pedersen and Fredenslund, and uses the Peng-Robinson (P-R) equation of state (Peng and Robinson, 1976). These numerical routines, contributed by Dr. Pin Du, an associate of Dr. Rao, are written in FORTRAN and require the temperature, pressure and composition of the fluid. The composition can be any combination of the 25 components (C_1, C_2, C_3, iC_4, nC_4, iC_5, nC_5, nC_6, nC_7, nC_8, nC_9, nC_{10}, N_2, CO_2, H_2S, toluene, benzene, cyclohexane, H_2O, H_2, CO, NH_3, CH_3OH, He, Bitumen) available and the corresponding mole fraction.

5.4.1.1 Peng - Robinson Equation of State (1976)

Before selecting a viscosity model, a fluid's thermodynamic state (i.e. liquid and/or gas) must be determined. For pure fluids this can be an easy task, but for more complex

mixtures it will be more difficult. Also, when calculating the viscosity using a model such as Dean and Stiel, the fluid's specific volume is required. Therefore, an equation state, such as the P-R, which is capable of reliably predicting this information is valuable.

The P-R is an equation of state capable of accurately calculating a fluid's thermodynamic state and physical properties (e.g. specific volume) for a wide range of pressures, temperatures and compositions (primarily hydrocarbon components). The equation is a two-constant equation of state of the form

$$P = \frac{RT}{(v-b)} - \frac{a(T)}{v(v+b)+b(v-b)} \tag{5.4}$$

where P is the absolute pressure, R is the universal gas constant in appropriate units; T is the absolute temperature, v is the specific volume (units of volume per mole); and $a(T)$ and b are values determined by the physical properties of individual components in the fluid. To calculate the values of $a(T)$ and b, the critical temperature and pressure, molar mass and Pitzer's accentric factor for each component along with binary action parameters, all contained in a database file within the program, are required.

In this thesis, this equation of state also provides the added value of being capable of predicting the density and compressibility factors. These values can be used by the flowmeter sizing programs that have no means to generate their own, such as MEM 1.0. Also, these values can supplement the other sizing programs such as the V-Cone 3.1 and FLOWEL 2.0 that are unable to calculate a density or compressibility factor for a certain fluid composition. In the case that both the flowmeter sizing program and the P-R can

calculate these values, the values predicted by the P-R can be used to validate the values calculated by the sizing program.

5.4.1.2 Viscosity models

Three viscosity models are used in this coupling system with each model complementing the other two. The models complement one another in that for a given fluid one model can more reliably predict the viscosity and/or more rapidly arrive at a solution than the other two.

Dean and Stiel (1965) presented a method to predict the viscosity of dense gases and high temperature liquids. In their method, the viscosity of a fluid can be determined by residual viscosity, $\mu_m^o - \mu_m$. The residual viscosity of the fluid is defined in terms of the fluid's reduced density, ρ_{rm}, as below:

$$\left(\mu_m^o - \mu_m\right)\xi_m = 1.08\times10^{-4}\left[\mathrm{Exp}\left(1.439\rho_{rm}\right) - \mathrm{Exp}\left(-1.11\rho_{rm}^{1.858}\right)\right] \qquad (5.5)$$

where μ is the viscosity (centipoise); ξ and ρ are variables dependent on the temperature, pressure and composition of the fluid; o indicates a reference state, m indicates a mixture, and r indicates a reduced property. In order to calculate μ_m^o, ξ_m and ρ_{rm}, pseudo critical constant mixing rules are employed using the critical temperature, pressure, volume and compressibility for each component in the mixture.

This model has been compared primarily with experiment data for hydrocarbon gaseous nonpolar binary and multicomponent mixtures. This model provides a fast convergent

scheme, but it is only valid for a fluid with $\rho_{rm} < 2$. Therefore, it cannot be used in a multicomponent liquid mixture.

On the basis of an extended corresponding states theory, Ely and Hanley (1981) developed a model to predict the viscosity of a non-polar fluid mixture. In their model, viscosity is defined by the following corresponding states model:

$$\mu_m(\rho_m, T) = \left(\frac{\Omega T_{cm}}{T_{co}}\right)^{\frac{1}{2}} \left(\frac{\theta V_{cm}}{V_{co}}\right)^{\frac{-2}{3}} \left(\frac{M_m}{M_o}\right)^{\frac{1}{2}} \mu_o\left[\frac{\rho_m \rho_{co} \theta}{\rho_{cm}}, \frac{TT_{co}}{T_{cm}\Omega}\right] \qquad (5.6)$$

where μ is the viscosity; ρ is the density; T is the absolute temperature; M is the molar mass; V is volume; o indicates a reference fluid at a very low pressure; c indicates a critical value, and m indicates a mixture value. The shape factors, Ω and θ, are functions of reduced volume that must be calculated by the same corresponding states model, equation (5.6). To predict the viscosity of a mixture, the mixture is first treated as a hypothetical pure fluid with a given T_{cm}, ρ_{cm} and M_m. Then, through the iterative technique, the shape factors will be calculated to determine the viscosity.

This model was tested on pure hydrocarbons from C_1 to C_{20} with an average deviation of about 8% and on a number of binary mixtures with an average deviation of about 7%. If the components differ substantially in their critical densities or if the temperature of a fluid is near its freezing point, this model cannot yield very satisfactory results.

Pedersen and Fredenslund (1987) also developed an algorithm to predict the viscosity of a complex mixture. Instead of using shape factors as in the model of Ely and Hanley, they proposed a rotational coupling coefficient, α, in the corresponding states model.

$$\mu_m(P,T) = \left(\frac{T_{cm}}{T_{co}}\right)^{\frac{-1}{6}} \left(\frac{P_{cm}}{P_{co}}\right)^{\frac{2}{3}} \left(\frac{M_m}{M_o}\right)^{\frac{1}{2}} \frac{\alpha_m}{\alpha_o} \times \mu_o\left[\frac{PP_{co}\alpha_o}{P_{cm}\alpha_m}, \frac{TT_{co}\alpha_o}{T_{cm}\alpha_m}\right] \qquad (5.7)$$

where the symbols are as defined previously.

This model has been compared with experimental data for pure and binary mixture hydrocarbons, and crude oils. It is capable of predicting both liquid and gaseous fluids. However, it is inferior to the model of Ely and Hanley in predicting viscosity of pure components and of binary mixtures.

5.4.2 Symbolic reasoning

In order to develop the coupling system, the commercial AI tool Personal Consultant™ Plus (Anon, 1987) from Texas Instrument Inc. was selected. The reason for selecting this tool was that it allows for rapid development of an expert system due to its numerous features. PC Plus, a LISP-based tool, offers features such as programming syntax error detection; a tracing technique for debugging; an easy to develop user's interface, on-line help and conclusion explanation; and a backward chaining reasoning mechanism. One key feature was that it could invoke an external language program such as the P-R and viscosity routines that are necessary for fast, efficient numerical computations.

The knowledge† is captured using frames†† , production rules and parameters. A frame contains information in two primary groups: rule groups and a parameter group. A frame may have one or more rule groups, but can only have one parameter group. Rules are the basic structures used to encode the knowledge and in the form of IF <condition> THEN <action> statements. In the premise and conclusion of rules, relationships among parameters are expressed in algebraic statements, and the symbolic logical of propositional and predicate calculus. Parameters are assigned either numerical or symbolic values.

Following is an example of a rule used in the decision frame which is shown in Figure 5.1:

IF <visc correlation is not definite>
Then <visc correlation = Pedersen and Fredenslund
and visc correlation index = 3>

If the premise is true, no value assigned to the parameter "visc correlation", then, in the conclusion, the parameter "visc correlation" is assigned the symbol "Pedersen and Fredenslund", and the parameter "visc correlation index" is assigned the numeral "3".

---

† Appendix C contains the knowledge as represented in PC Plus.

†† The term frames were used in the PC Plus instruction manual to describe a collection of groups. This is unfortunate choice of terminology since the frame structure in PC Plus differs significantly from the frame-based knowledge representation in systems such as KEE and Meta-COOP which is usually associated with the term frame. However, for the remainder of this section, the term frame will be used to describe the knowledge representation structure used in PC Plus.

In this coupling system, the knowledge base is organized into a three frame hierarchy (input, decision and collection, in respective order from top to bottom) with multiple rule groups. Although this sectioning is arbitrary, it facilitates programming since the smaller sections are easier to understand than one large section. Also, in most cases, these sections can be modified without affecting the other sections.

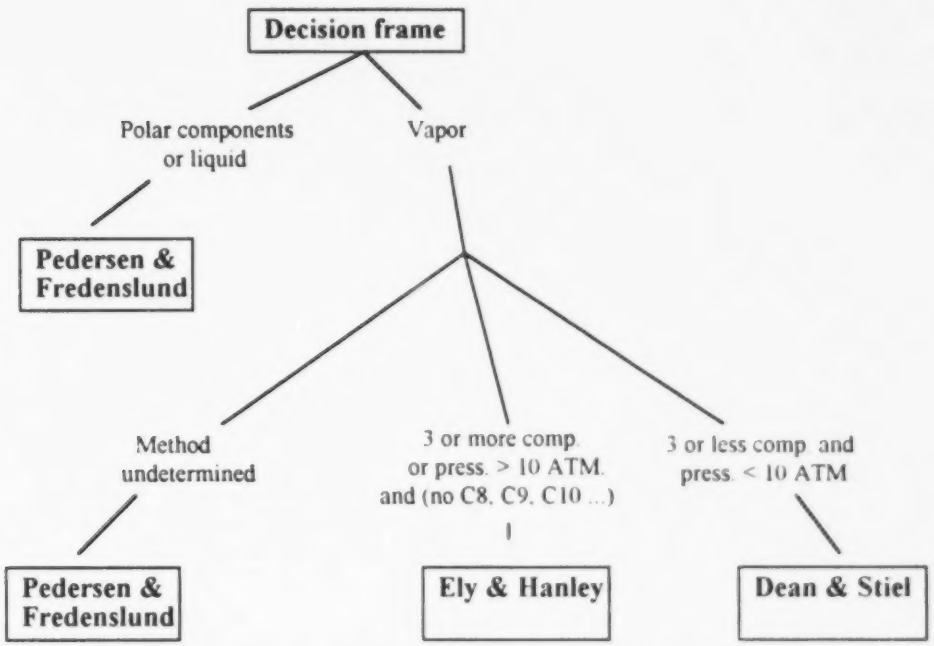
Decision frame
Polar components or liquid
Vapor
Pedersen & Fredenslund
Method undetermined
3 or more comp. or press. > 10 ATM. and (no C8, C9, C10 ...)
3 or less comp. and press. < 10 ATM
Pedersen & Fredenslund
Ely & Hanley
Dean & Stiel

Figure 5.1 Decision frame knowledge organization

The input-frame obtains information required for the selection of a viscosity model and transfers this information to other areas of the program. The frame is divided into three rule groups. The units-rule group prompts the user to select the system of units preferred, and to enter the temperature and pressure in these units. The input-rule group prompts the user to identify the fluid's components and the mole percentage of each component. The

write ext-rule group transfers information collected to a data file and invokes the P-R equation. The P-R equation predicts the fluid's thermodynamic state.

The decision-frame is divided into two rule groups. One rule group allows the user to choose the desired model. This is implemented so that the different models can be compared with each other or an external source. The other rule group selects the most appropriate model based on the fluid's characteristics such as the individual components present and whether the fluid is liquid and/or vapor. The knowledge used to determine the most appropriate model for a given fluid was obtained through discussions with Dr. P. Du.

The collection-frame helps the decision-frame collect information determined throughout the consultation and displays the results. Because the tasks performed by this frame are simple, there is only one rule group.

For PC Plus to interface with the numerical routines, data files are used as shown in Figure 5.2. This required PC Plus to place information in the files in a form the numerical routines could access and interpret. In order to accomplish this task, the methods used by the numerical routines to collect input has to be altered. Also, the method that the numerical routines use to display results has to be changed to a form PC Plus could understand.

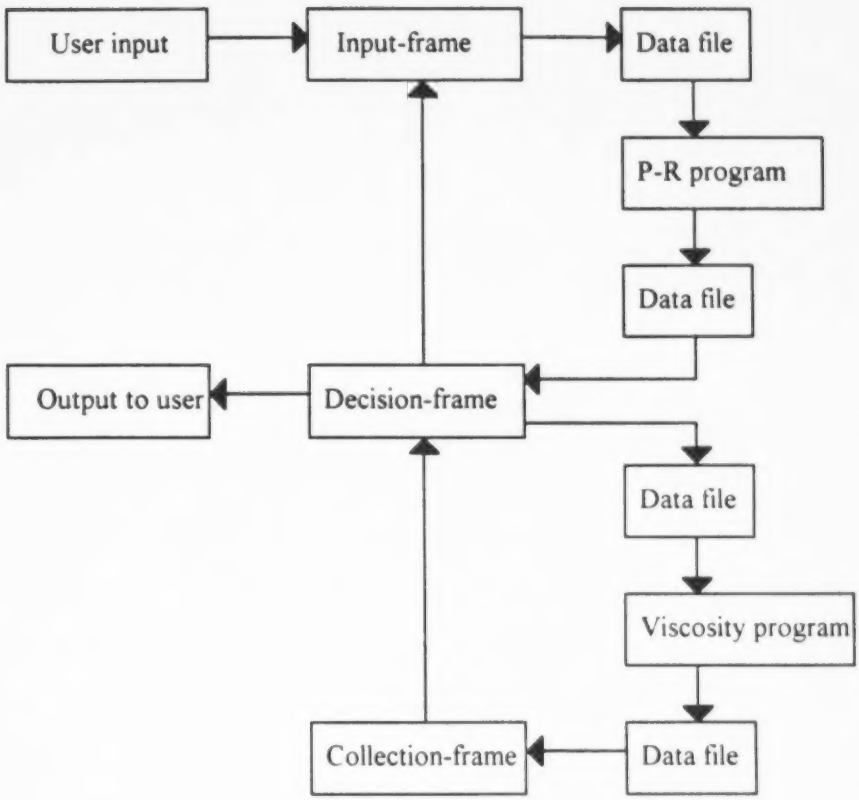

Figure 5.2 Schematic diagram of information flow in the coupling system

Chapter 6

System Implementation

In developing an IDIS for differential pressure flowmeter selection and sizing, there are two major obstacles: one is to integrate the individual software packages into an integrated system, and the other is to organize the meta-system knowledge base for selecting and coordinating these individual activities. In this chapter, the technique of integrating the individual programs, building and organizing the content of the meta-system knowledge base is presented.

6.1 System Integration

Since Meta-COOP version 2.0 is written in C and the source code is available, the program could be used on any computer system with a C compiler. However, this version is primarily developed to be run on a computer with a UNIX® operating system and the X Window System™. With Meta-COOP running on a UNIX operating system and the flowmeter sizing software as well as viscosity coupling system compiled for DOS, the challenge is to combine both the heterogenous operating systems and the application software into a single integrated system.

Before finally choosing a means to integrate these heterogenous software packages, a number of alternatives are considered. One alternative is to re-write all the sizing

programs and coupling viscosity system so that the source code is available. Then, compile the source code on a UNIX operating system. This alternative is not viable, however, because the coupling viscosity system would have to be reconstructed using a different development tool since PC Plus is not known to be available for UNIX, the V-Cone sizing equations are unavailable; there is already good commercial software for sizing differential flowmeters available; and finally, this would invalidate a primary concept of IDIS which is to use existing software in an unaltered form.

Two other alternatives are also considered. One involves writing a number of routines and using a serial connection between the UNIX-based computer and the DOS-based computer. The other requires installing the OS/2™ operating system on a PC and using a network connection between the UNIX machine and PC. Although it may have been possible to accomplish the objective of system integration using either one of these alternatives, it would not have satisfied another objective. This objective is to have input and output from all programs displayed on the same monitor.

In order to achieve this last objective, two other alternatives are considered. One alternative is to install DOS emulation hardware and software on the UNIX machine. Although this option would satisfy all objectives and eliminate the need for the PC, the cost is prohibitive. As a result, another alternative is chosen. This alternative involves installing a commercial software package DESQview/X™ on a PC running DOS and utilized a network connection between the UNIX machine and PC.

Figure 6.1 presents an overview of IDISDPFSS including the computer programs involved and which hardware platform and operating system they run on.

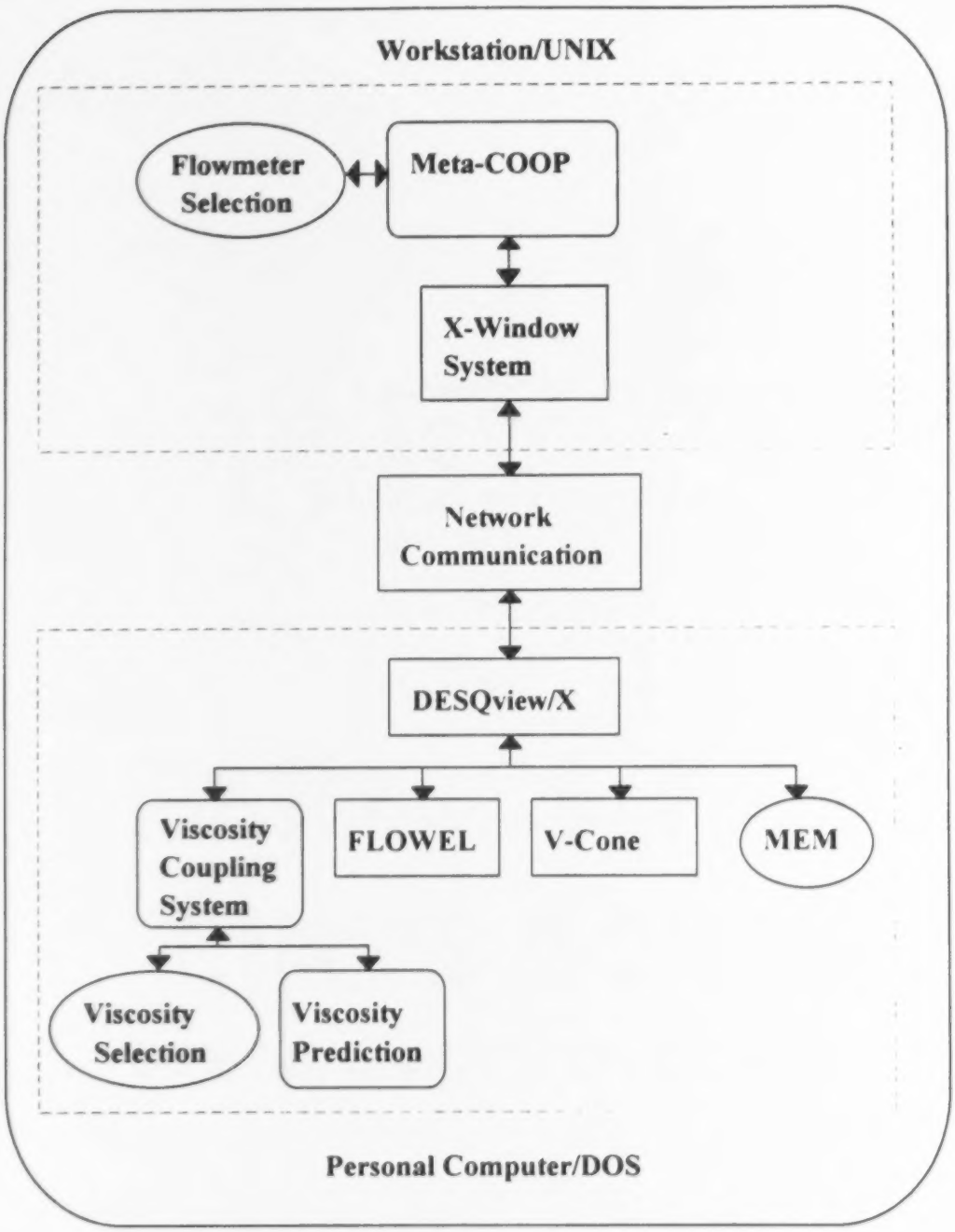

Figure 6.1 Overview of IDISDPFSS integration.

In the following three sub-sections, the equipment used, software/operating systems involved and introductory concepts in network communications and the X Window System will be presented. The first sub-section will deal with the UNIX machine while the second sub-section will describe the DOS PC. A third sub-section will present detailed information on the communication implementation.

6.1.1 Hardware, software and UNIX operating system

Meta-COOP is run on a Sun® SPARC™ 1+ workstation (computer) within the Intelligence Engineering Laboratory in the Department of Chemical Engineering at the University of Alberta. This computer has 24 megabytes of random access memory (RAM) and a one gigabyte hard drive. The monitor is a 16 inch color graphic display. The operating system is SunOS™ 4.1.3. This operating system uses the transmission control protocol/internet protocol (TCP/IP) network protocol. It is a UNIX type and contains OpenWindows™ version 3.0. OpenWindows is an environment which contains a Graphic User Interface (GUI) based on the X Window System. The "look" of the interface is designed to the OPEN LOOK™ specification.

SunOS 4.1.3, like all UNIX type operating systems, is designed to be multi-tasking. A multi-tasking operating system allows different programs/processes to run concurrently. As an example, a user may run any numerical computation program, while editing a text file. In contrast, a single task operating system such as DOS allows only one program at a time to run. In other words, a program must be run until it is finished before another program can be initiated. Using the above example, the numerical computation program would have to be terminated before the user could start editing the text file.

"The X Window System is hardware-independent and operating system-independent graphics standard designed to operate over a network..." (Radcliffe, 1992) in a multi-tasking operating system. As an example, the X Window System is capable of running on hardware architectures such as HP 9000, IBM R6000 and, in the present study, Sun SPARC 1+ station and IBM compatible PC. Also, the X Window System operates with the corresponding multi-tasking operating systems: HPUX, IBM AIX and, in this implementation, SunOS 4.1.3 and DOS with DESQview/X.

The X Window System provides the capability to develop a GUI through which information can be passed between the user and program. GUIs usually provide nicer looking interfaces and are easier to work with when supported by a mouse than the traditional text or character-based interfaces.

In order for the X Window System to display the user interface of any software package, the system must be able to interpret all commands the package uses to display information on the monitor. In some cases, these software packages employ special graphic routines to display information on the monitor, which the X Window System is unable to interpret. As a result, these software packages cannot be used in conjunction with the X Window System. This is why the windowing package chosen to develop MEM 1.0 could not use special graphic routines.†

A network is a configuration of computers that are connected for the purpose of passing information between themselves. For these machines to successfully pass information,

---

† For a more complete presentation of the X Window System employed on the PC in this work, the reader is referred to Radcliffe (1992).

they must communicate in a way that is understandable by many different types of hardware architectures and operating systems. As a result, network protocols such as TCP/IP have been developed, which are independent of hardware architectures and operating systems. These protocols define a convention to be followed (Anon, 1990).

Network protocols differ from communication packages such as Kermit in that they allow multi-tasking machines to pass information among the various tasks running concurrently. In contrast, a computer running Kermit without a network protocol† is limited to passing information only through Kermit. The advantage of passing information among various tasks running simultaneously will be illustrated in Chapter 7.

6.1.2 Hardware, software and DOS

The flowmeter sizing programs and viscosity coupling system are run on an IBM compatible PC. The central processing unit (CPU) is a 486 operating at 33 megahertz. The computer has 8 megabytes of RAM, a 210 megabyte hard drive and is connected to the network by an Ethernet card plus elite 16 Series and TCP/IP. The operating system, as previously mentioned, is DOS.

Since DOS is only a single tasking operating system and a multi-tasking environment is required in order to utilize a network protocol and the X Window System, DESQview/X, a commercially available software package, was installed on the PC. DESQview/X runs

---

† On a multi-tasking machine with a network protocol, one of the tasks that may be running is a communication package such as Kermit.

on top of the DOS and converts the PC from a single tasking operating system into multi-tasking environment with the added advantage of providing the X Window System.

DESQview/X is analogous to the more familiar Microsoft® Windows™ except that the GUI in DESQview/X is the X Window System rather than the GUI developed by Microsoft. DESQview/X consists of three components: the Quarterdeck expanded memory manager-386 (QEMM-386™) which controls the expanded RAM; the DESQview package which works with QEMM-386 to provide the multi-tasking environment; and "/X" which provides the X Window System capability. In addition to this software, other software required is the DESQview/X network manager to other X Systems TCP/IP option and Novell® TCP/IP Transport for DOS.

While the above software combination provides many advanced features†, the key applicable feature to this thesis is that the IDISDPFSS is able to initiate the flowmeter sizing programs and the viscosity coupling system on the PC, yet display these programs' interfaces on the Sun monitor which is connected to where the IDISDPFSS is running. Also, once the PC programs are running, they can receive information entered from the Sun workstation.

6.1.3 Remote execution of programs

In order to make the execution of the PC programs transparent to the user, the process of remotely executing and locally displaying these programs has been automated. This has been accomplished by configuring database files on both the Sun SPARC station and PC.

---

† The interested reader is referred to Radcliffe (1992) for a complete description of all available features.

In this section, the database files that must be configured and the commands to execute the PC programs are described.

In network communications, when two computers wish to exchange information, the computer address of each computer must be specified. An address is of the form xxx.xxx.xxx.xxx where each group of xxx is a number from 0 to 255. Since this address may be difficult to remember, the computer address can be associated to a name. This is accomplished by an entry placed in the database file "hosts". An example of an entry in a host file, on the Sun computer, is given as follows:

129.128.56.30 rao5

Thus, a user can specify rao5 as the computer to communicate with rather than 129.128.56.30.

In order for a program run on a remote computer to use the initiating (local) computer's monitor, permission must be granted by the local computer. This permission is given by executing the command "xhost". In the current case-study, the command "xhost +rao5" is automatically issued prior to entering the IDISDPFSS on the Sun computer.

One UNIX command (there are others) to initiate a program on a remote machine and display on the issuing machine's monitor is "rsh". This command is chosen because it requires only one line of input and a minimum number of parameters. The form of the command is

rsh HOST -l USER Program_name

where HOST is the name of the remote computer; -l specifies that the following argument is the name of a "USER"; and "Program_name" is the name of the program to initiate. In this case-study, the HOST name is rao5 and the USER's name is "murray". The USER's name has been specified in a database file on the remote machine and not necessarily the name of the person running IDISDPFSS. The Program_name is the name of the program to be executed on the HOST computer. As an example, to execute the V-Cone flowmeter sizing program on the PC from the Sun computer the following command has to be entered:

```
rsh rao5 -l murray vc310000 dvp
```

For this command to be successfully received on a machine running DESQview/X the rsh option must be enabled.

In the above example, the program vc310000 dvp is not the actual name of the program executed on the PC but one assigned by DESQview/X's DVP (DESQview program) manager. The DVP is used to create an information file that contains system requirements for each DOS-based program required. This information file enables DESQview/X to run the program efficiently (Anon., 1993) and provides the additional advantage of being able to execute the program directly from another machine rather than issue an extra command.

6.2 Building a Knowledge Base

When building a knowledge base, no matter whether it is for a meta-system or expert system, there are two types of knowledge that can be employed. One type is public knowledge (Rao and Qiu, 1993). This knowledge is contained in books and articles, and readily accessible to the public. As an example of this in flowmeter selection, it is generally stated that flow nozzles can only be used with two inch or larger nominal diameter pipe (Ginesi, 1991; Danen, 1985).

The other type of knowledge that can be employed is private knowledge (Rao and Qiu, 1993). This knowledge is possessed by experts in a field. This type is applicable to solution of a real-world problem. Using the above flow nozzle example, although flow nozzles can be manufactured for two inch nominal diameter pipes, in most cases, they are only practical for pipes six inches and larger because of the difficulty in locating the pressure taps.

When building a knowledge base, personal experience can be applied to simplify the solution of a problem. While the simplified solution is less accurate than that obtained by more rigorous methods, the time required to reach it is reduced. An example, in flowmeter selection, the amount of straight pipe length upstream and downstream of the flowmeter required to obtain an accurate measurement depends on the pipe fittings and their configuration upstream and downstream of the variable differential pressure flowmeter (Miller, 1983). However, it is generally accepted that straight pipe lengths of 10 pipe diameters upstream (Hasley, 1986) and 5 pipe diameters downstream are sufficient.

Another point to consider when building a knowledge base is the amount of time required to search the problem space before reaching a solution. In order to reduce this time, heuristics are employed. Heuristics are the shortcuts, rules of thumb or strategies that experts use to reduce the time to reach a solution. Heuristics can be employed in two ways. One way is not to include irrelevant knowledge. For example, if designing an IDISSDPFSS for use in North America, ASME flow nozzles need only to be included as they are the mostly commonly used in North America. However, if the IDISDPFSS is intended for use in Europe, ISA flow nozzles should be included and ASME flow nozzles excluded. The other approach is to use heuristics to reduce the problem space as quickly as possible based on the available information. In IDISDPFSS, this is accomplished by performing an initial check based on the user's requirements to determine if any of the flowmeters are applicable.

From this presentation, the reader might gain the impression that public and private knowledge, experience and heuristics and their utilization in IDISDPFSS may not always provide the appropriate solution. And the reader would be correct. However, for the majority of applications, the IDISDPFSS will select the most appropriate flowmeter based on a limited number of specifications.

6.3 Selection Methodology

The methodology employed by the IDISDPFSS is based on the general two stage procedure presented in Section 2.4. Using this methodology, all the flowmeters are initially considered appropriate for each application. Then, on the basis of the user's requirements, inappropriate flowmeters are eliminated. This is opposed to initially considering no flowmeters as applicable and then selecting flowmeters that are

appropriate. The former method is preferred because one factor can eliminate a particular type of flowmeter but many factors must be considered to determine if a flowmeter is applicable. As an example, if a fluid contains particulate matter, the orifice plate is an unsuitable candidate and can be disqualified. However, based only on the information that fluid contains particulate matter, the V-Cone, which can tolerate particulate matter, cannot be considered applicable until other factors are investigated.

6.4 Knowledge Base Organization

The meta-system knowledge base for IDISDPFSS contains over 50 memberslots and 50 rules, as given in Appendix A. These memberslots and rules are distributed into eight knowledge bases to facilitate maintenance and modifiability by providing smaller sections that are easier to understand than one large knowledge base. Also, each knowledge base contains only one unit and one method, thus minimizing any confusion about where a message is being sent.

Of the eight knowledge bases, only the flow_req.kbs is not directly associated with the other knowledge bases. The flow_req.kbs contains all the user's requirements used in the selection process. The user's requirements are contained as memberslots and therefore are available to all the other knowledge bases.

The other knowledge bases are arranged using decomposition. This organization takes advantage of the graphical display capabilities of Meta-COOP as will be illustrated in Chapter 7 and is used to facilitate maintenance and modifiability. Ease of maintenance and modifiability is achieved by distributing specific functionality into different knowledge bases. As an example, the knowledge for determining which of the four flowmeter types

are applicable is contained in four separate knowledge bases, one for each type of flowmeter. Therefore, if one wishes to change the selection criteria for a specific flowmeter type, only one knowledge base need be altered.

The top or root knowledge base, flow.kbs, is decomposed into six knowledge bases as shown in Figure 6.2. The method slot in flow.kbs is where the first message is sent when the process of meter selection is invoked by the user. Flow.kbs is the controlling knowledge base and is responsible for sending messages to the other knowledge bases. After each of these knowledge bases has finished its processing, control is returned to flow.kbs. Using Figure 6.2 in order to illustrate this control scheme, a message is sent from flow.kbs to flow_check.kbs which upon completion returns control to flow.kbs. Flow.kbs then in-turn sends a message to flow_var_area.kbs and so on. By passing control in this manner, if a new flowmeter type is to be added to the selection process, only one message statement has to be added to flow.kbs.

The first knowledge base that flow.kbs sends a message to is flow_check.kbs. The function of this knowledge base is to determine whether any of the flowmeter types are applicable. This knowledge base differs from the knowledge bases containing selection criteria for the individual flowmeters in that if the user's requirements are greater than the specifications in this knowledge base, the selection process terminates without proceeding to the other knowledge bases†. As an example, if the nominal diameter pipe size is less than 1/2 inch, the selection process is terminated because none of the meters are generally

---

† It is possible for the selection process to proceed to the individual flowmeter knowledge bases but determine that no flowmeter is applicable.

manufactured in a size smaller than this. This knowledge base saves the user's time by not searching the other knowledge bases if no flowmeter will meet the requirements.

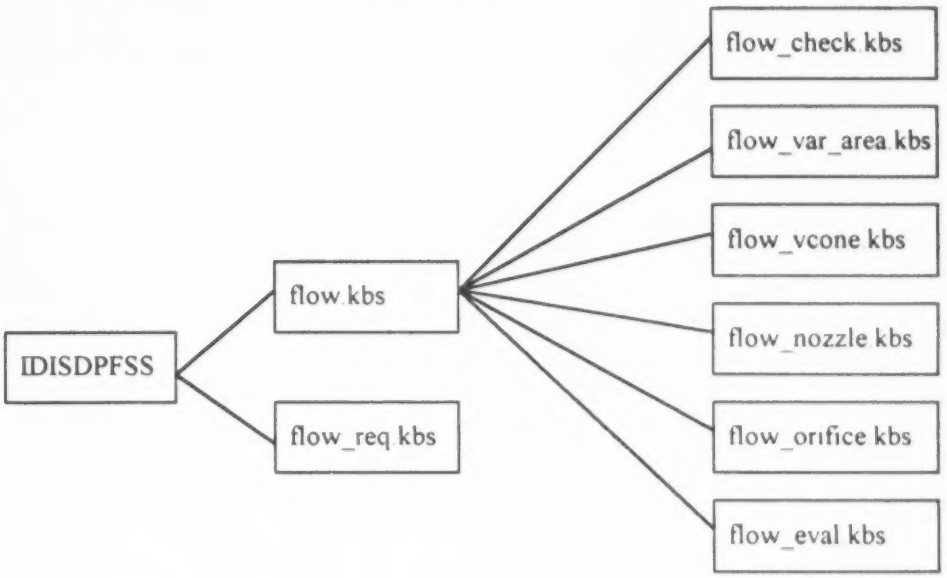

Figure 6.2 Illustration of meta-system knowledge organization for IDISDPFSS

The flow_var_area.kbs, flow_vcone.kbs, flow_nozzle.kbs and flow_orifice.kbs knowledge bases each contains criteria to determine whether the particular flowmeter type will meet the user's requirements. Each knowledge base contains four memberslots. Two of the memberslots store the result of the selection process with one slot used to display the result graphically for the user as presented in Chapter 7. One other memberslot contains the method to which the message is sent from flow.kbs. This slot invokes the processing of the symbolic knowledge contained in the rule slot to determine whether the specific flowmeter type is applicable based on user's requirements. The number of rules contained in the slot varies with the type of flowmeter. The knowledge bases for the flowmeter

types that have a narrower scope of applicability such as the variable area and orifice plate contain more rules than the knowledge bases for the more generally applicable flow nozzle and V-Cone. The number of rules contained in the variable area, orifice plate, flow nozzle and V-Cone knowledge bases are 17, 9, 8 and 6, respectively.

The knowledge base flow_eval.kbs determines the ranking of the flowmeters deemed applicable during the selection process. When developing this knowledge base, many factors related to the cost of employing a particular flowmeter in a specific application are considered. For example, it may be asked if the flow nozzle and V-Cone have a broader range of applicability and can generally be used in place of the variable area and orifice plate flowmeter, then why should one consider these latter two devices. One reason is that the initial cost of a variable area or orifice plate is less than that for flow nozzle or V-Cone for the same application. In general, for the four flowmeter types, with respect to initial cost, the order from lowest to highest is the MEM variable area meter, orifice plate, flow nozzle and V-Cone. It might be argued that an orifice plate is less expensive than a variable area flowmeter but when the cost of ancillary equipment such as valves, flanges and a differential pressure indicator are included, the cost of the orifice plate is more expensive. Another reason is that operating costs such as maintainability, reliability and permanent pressure loss must be considered. However, when comparing the MEM variable area, orifice plate, flow nozzle and V-Cone, these can be considered negligible. As for maintainability, because of the MEM flowmeter construction, it requires little maintenance. Also, while the orifice plate generally requires more maintenance, due to wear on its sharp edge, than the flow nozzle and V-Cone, the cost of this maintenance has been minimized by not selecting the orifice plate for harsh (particulate matter, erosive or corrosive fluids) applications. With regards to reliability, all four flowmeters are similar – all of them are not subject to sudden failure. As for permanent pressure loss, the variable

area meter is generally the lowest of the four meter types with the orifice plate, flow nozzle and V-Cone generating about the same permanent pressure loss for the same generated differential pressure†.

On the basis of the above information and with the objective of lowest overall cost, the flowmeters in flow_eval.kbs are ranked in order of preference as follows: MEM variable area, orifice plate, flow nozzle and V-Cone.

---

† Although intuition would suggest that the flow nozzle, because of its contoured shape, should cause much less permanent pressure loss than the orifice plate, this is not correct. The reason is that for the flow nozzle to generate the same differential pressure at the same flowrate as the orifice plate, the flow nozzle passes the fluid through a smaller aperture than the orifice plate. Consequently, the flow nozzle and orifice create about the same permanent pressure loss (Miller, 1983).

Chapter 7

IDISDPFSS Interface and Demonstration

The IDISDPFSS user/developer interface is menu driven and customized from a more general Meta-COOP interface. The customization involves the deletion of some inappropriate buttons and the addition of some new buttons specific to the process of flowmeter selection and sizing. The buttons that are added must be connected to the desired functionality which requires modifications to Meta-COOP's source code. The resulting main menu is illustrated† in Figure 7.1.

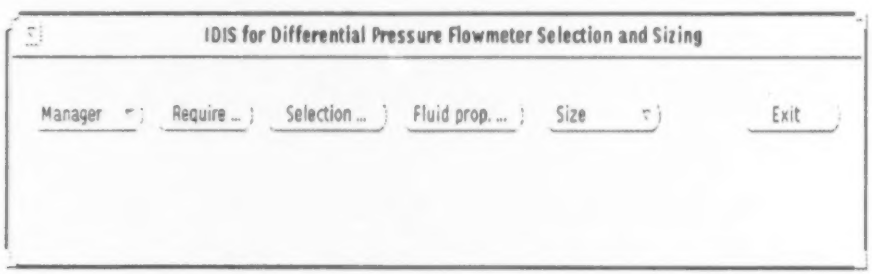

Figure 7.1 IDISDPFSS main menu.

The main menu consists of six buttons. The buttons are activated by placing a mouse driven pointer on them and pressing either the left or right mouse button. When pressing the right mouse button, the buttons "Manager ∇" and "Size ∇" display a pop-up menu

† All figures in this chapter are hard copies from the Sun SPARC station monitor.

which contains additional buttons. When pressing the left mouse button, the buttons "Require ...", "Selection ..." and "Fluid prop. ..." display a window which will require a user's response. When pressing the left mouse button, the "Exit" button performs an action without further user input.

The "Manager" button is intended to be used by the meta-system knowledge base developer and is standard function of Meta-COOP. This button displays a pop-up menu containing two other buttons: "File Manager" and "KBS Manager". The "File Manager" button displays a window that allows the developer to load and edit the knowledge bases (ASCII files). The KBS Manager button displays a window that allows the developer to load different knowledge bases and compile them.

The "Require" button displays the requirements which the user enters before proceeding to the flowmeter "Selection" process. Figure 7.2 shows the nineteen requirements which are displayed using a scrolling window. The requirements can be entered manually or loaded from a file. A file, which has the extension req, can be chosen from the "Require File Name List" by positioning the pointer on the name and clicking the left mouse button. Alternatively, a file name, not required to have the extension req, can be typed into the "File Name" area. In either case, the file is loaded by positioning the pointer on the "Load" button and pressing the left mouse button. Once a file has been loaded, the requirements can then be edited by positioning the pointer on a requirement's "Value " area and typing the appropriate data. To assist the user, each requirement states the "Datatype" expected: string, integer or real. In the case of string data types, a list of expected values is given next to the area marked "Choices". If beside the "Choices" area a ∇ exists, an appropriate value can be selected by placing the pointer on the ∇ and

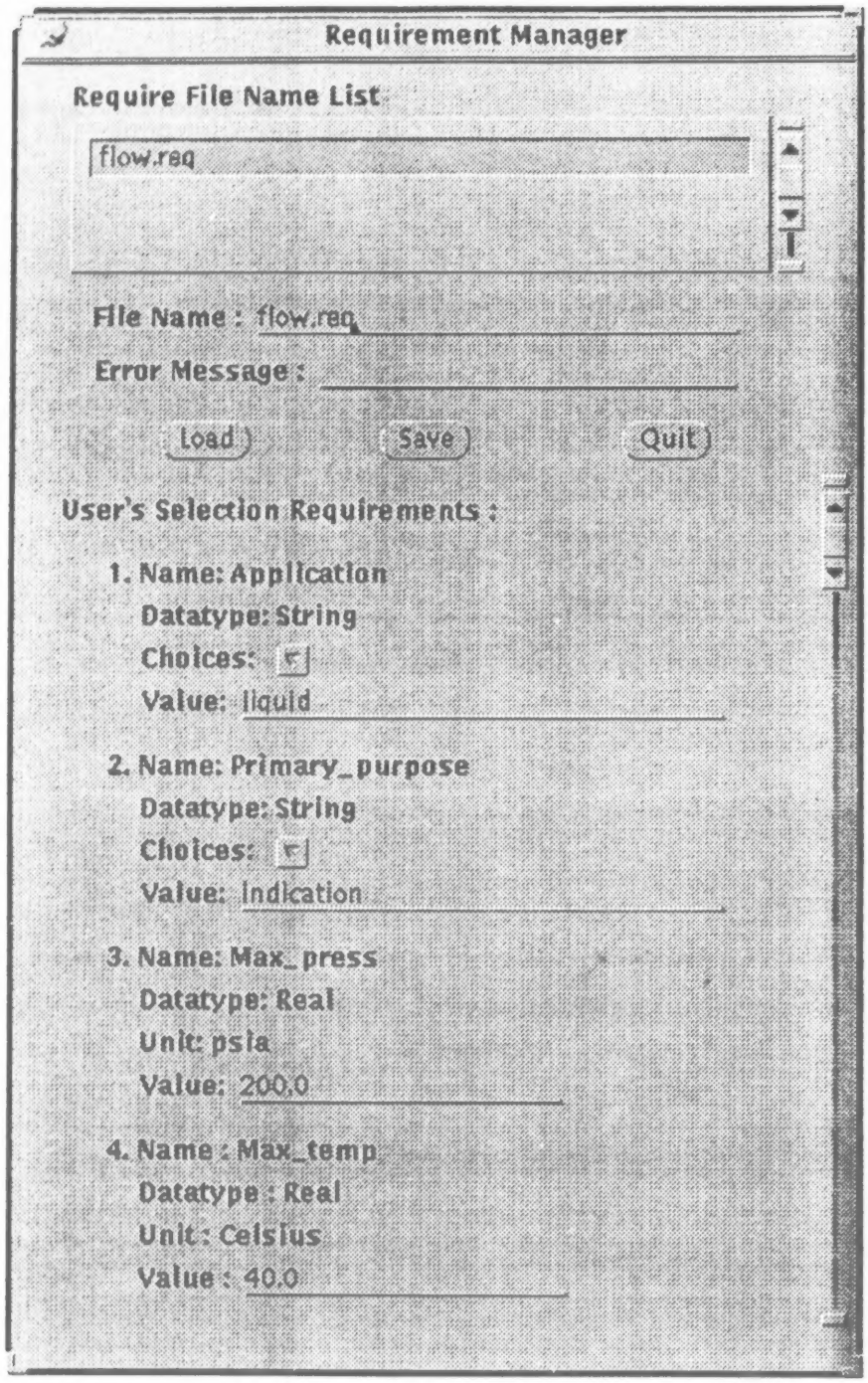

Figure 7.2 User defined requirements.

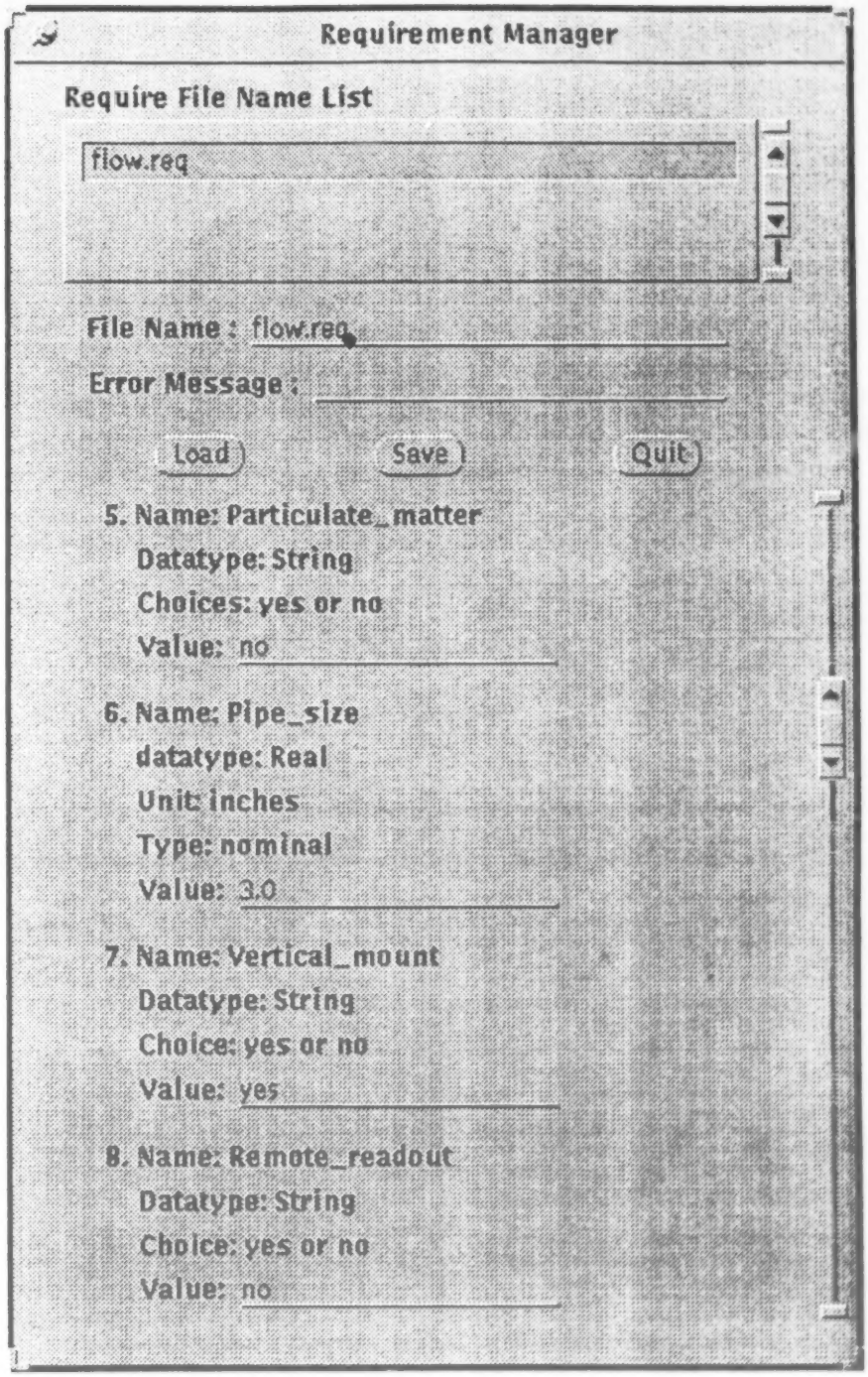

Figure 7.2a Continued.

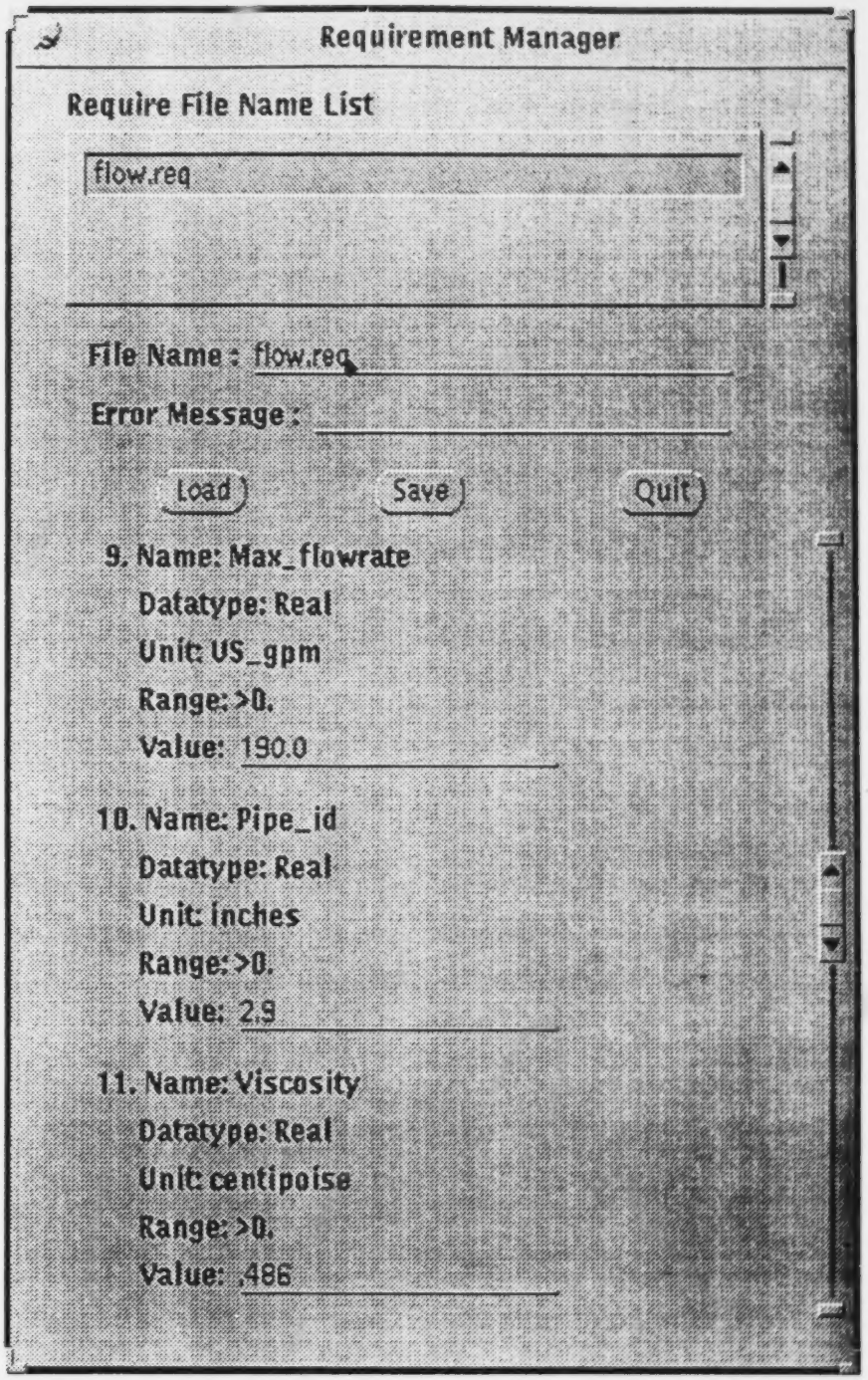

Figure 7.2b Continued

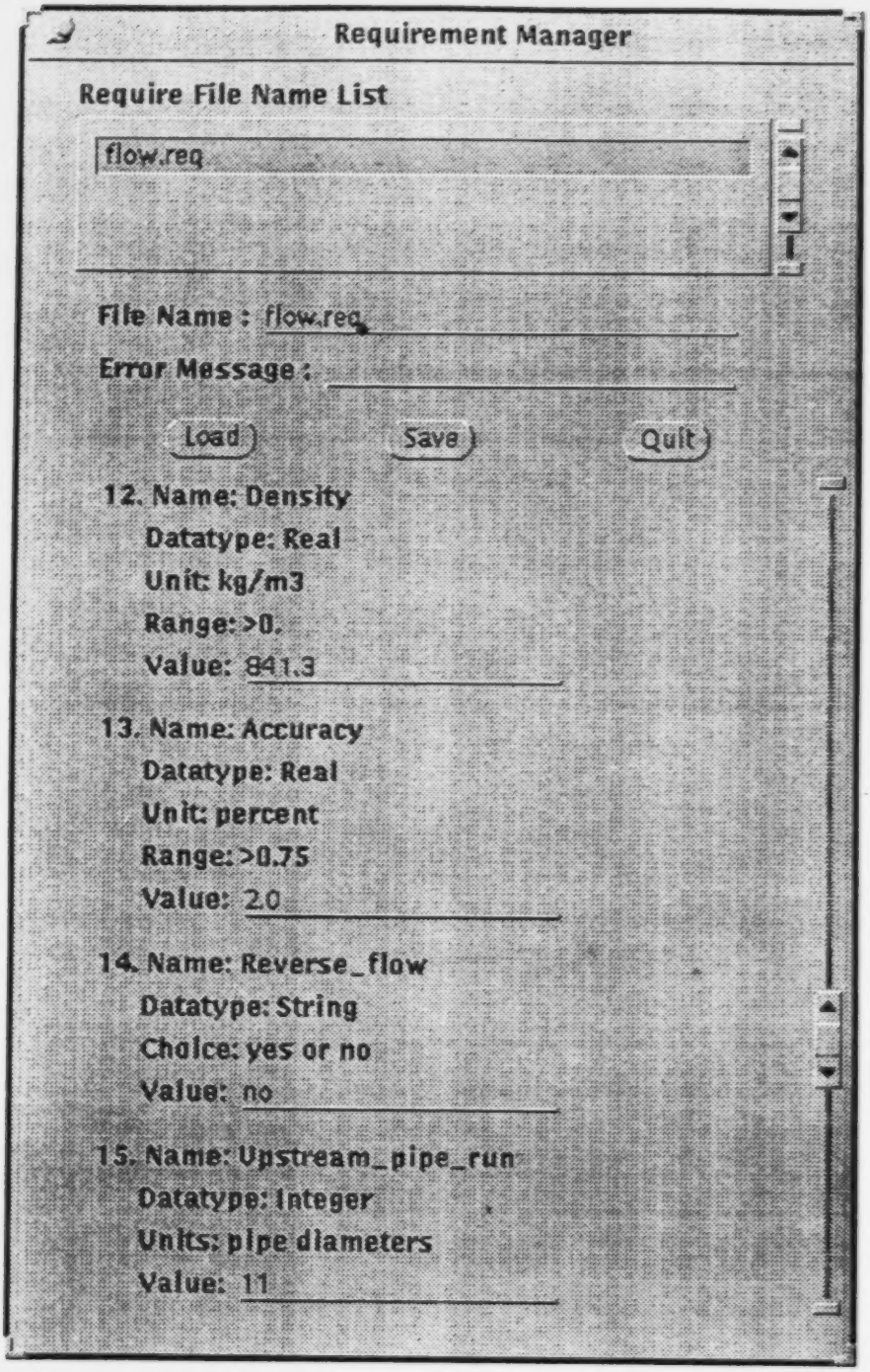

Figure 7.2c Continued.

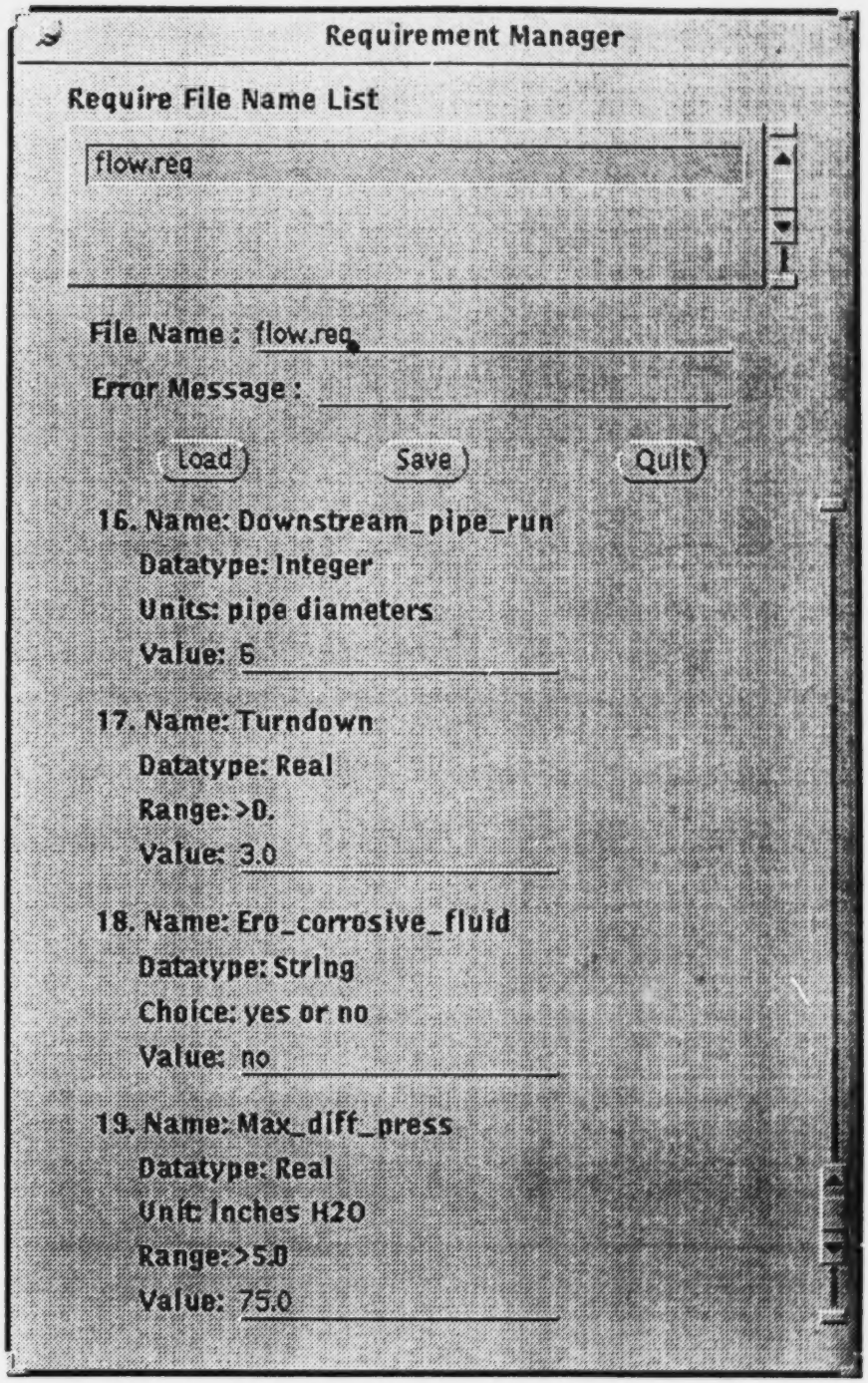

Figure 7.2d Continued.

pressing the right mouse button. This will display a pop-up menu with a list of values from which to select as shown in Figure 7.3. In the case of "yes" or "no" values beside the "Choices" area, the value is entered by typing either "yes" or "no" next to the "Value" area. To assist the user with integer and real data type requirements, each requirement states the "Units" in which the value is to be entered and in many cases a range of acceptable values is shown. For integer data types no decimal point is included and for real data types a decimal point must be included. No matter how the requirements are loaded, the information must be transferred to the "flow_req.kbs" knowledge base by positioning the pointer on the save button and pressing the left mouse button. The "Quit" button removes the requirement display.

After completing the requirements, the process of selecting an appropriate flowmeter is initiated by placing the pointer on the "Selection" button on the main menu and pressing the left mouse button. This action also causes the window shown in Figure 7.4 to be displayed. The area in the top right displays the results of the intermediate steps in the selection process. Interpreting Figure 7.4, the memberslot "req Reynolds_no" has been assigned a value of 119463.722812† as displayed in the statement "(1) req Reynolds_no = 119463.722812". The statement "Rule 42 has done"†† indicates that this assignment is performed by rule 42 in the knowledge base. At this point, the user may display the statement that determines this value, alter the value itself or proceed with the selection process. To display the statement, the number to the left of the assignment statement, in this example "1", is typed into the area right of the "Explain/Modify Item:", located at the bottom right of Figure 7.4, and the "Why" button activated by positioning the pointer on

---

† The author has no control over the number of significant digits displayed.

†† The author has no control over the wording of these statements.

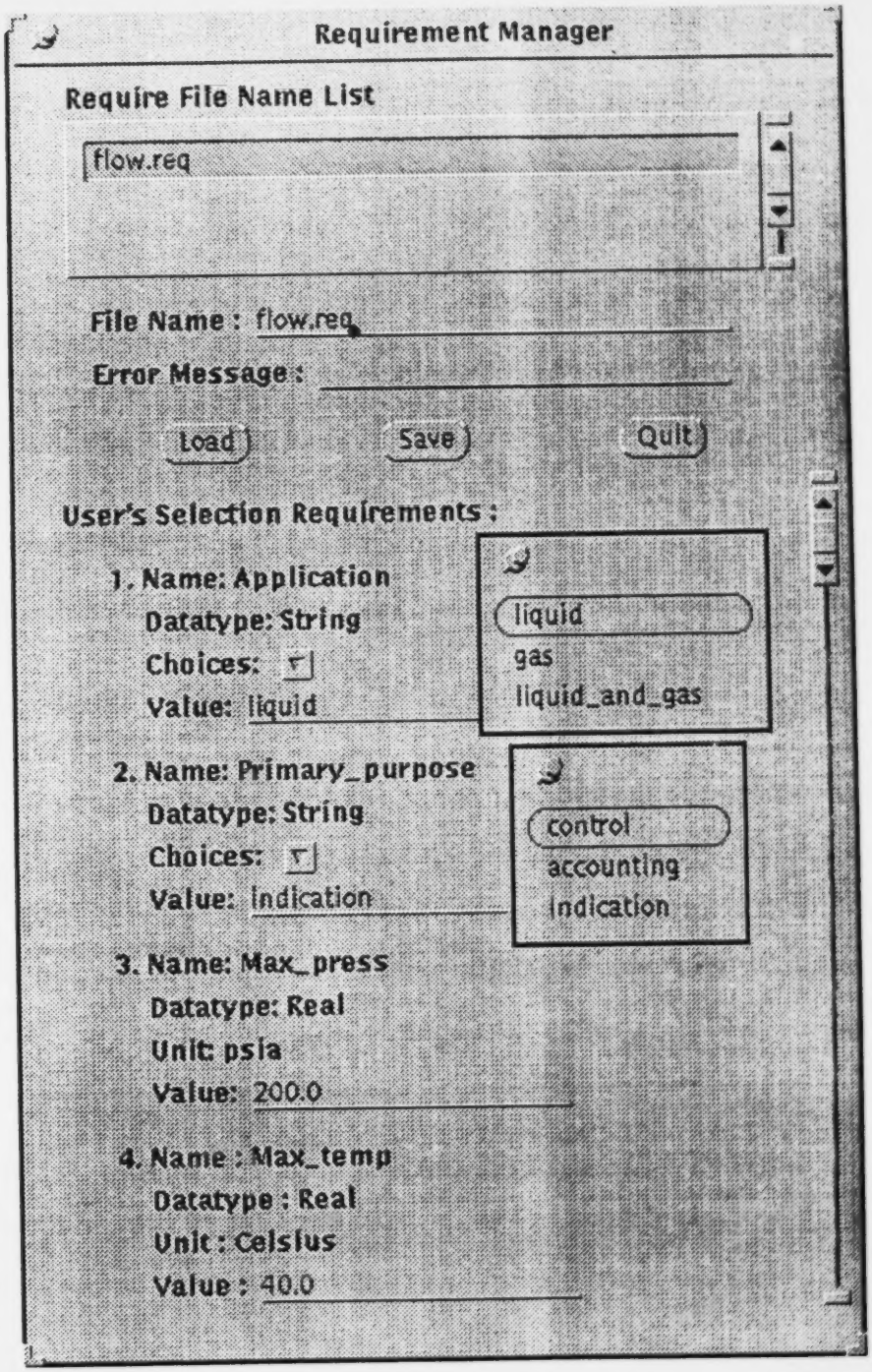

Figure 7.3 "Choices" pop-up menu.

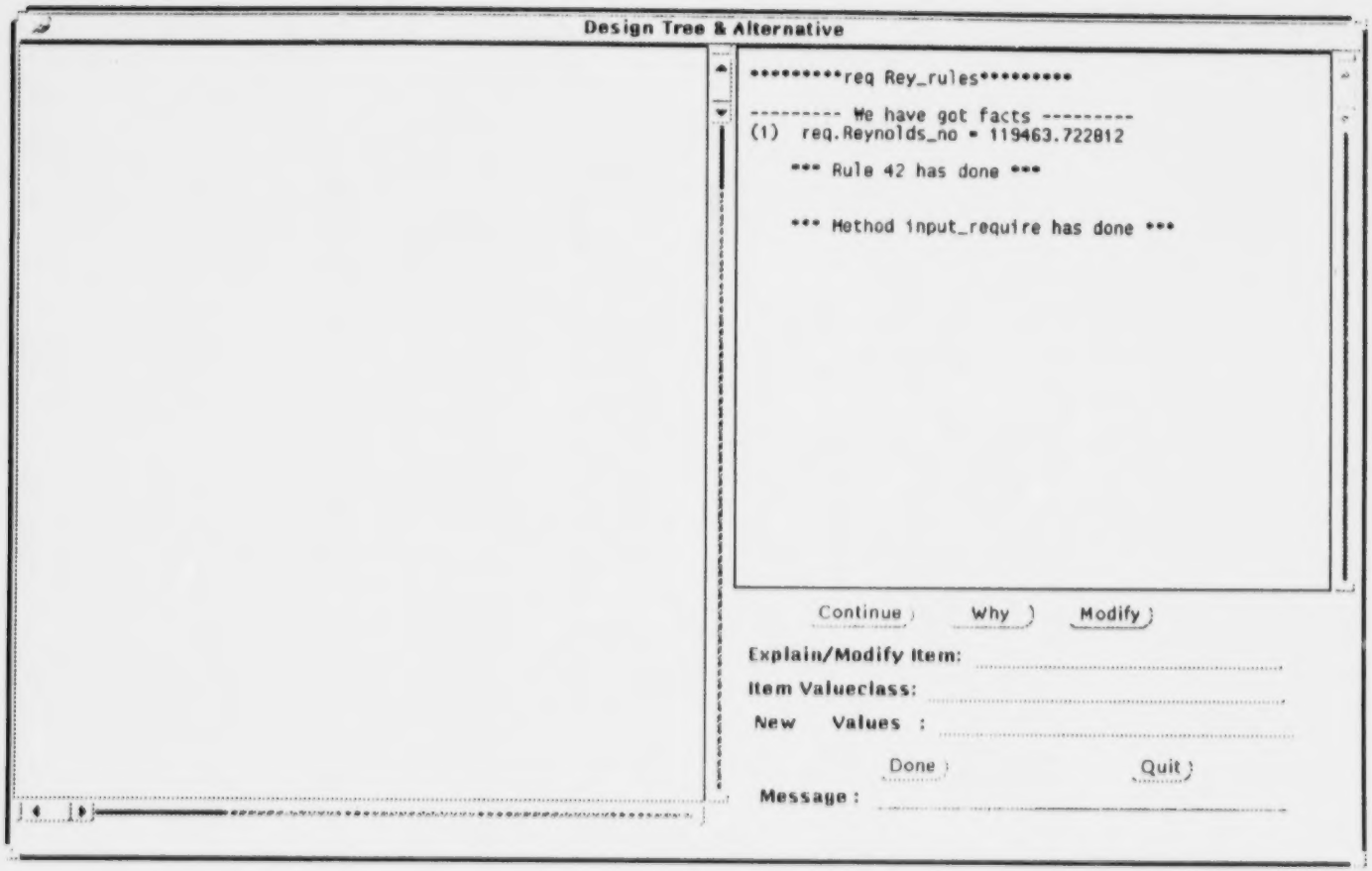

Figure 7.4 Selection process window.

this button and pressing the left mouse button. This results in the statement that determines the value of the memberslot to be displayed, as shown in Figure 7.5 for rule 42. To alter the value, the user types the number to the left of the assignment statement in the "Explain/Modify Item:" area and activates the "Modify" button. This results in the value class of the item being displayed in the area to the right of "Item Valueclass:". The user is then required to enter a new value into the area right of the "New Values :" and activate the "Done" button. The user proceeds with the selection process by activating the "Continue" button. The statement "Method input_require has done" below the "Rule 42 has done" statement indicates that a message is sent to this method and the method has been completed.

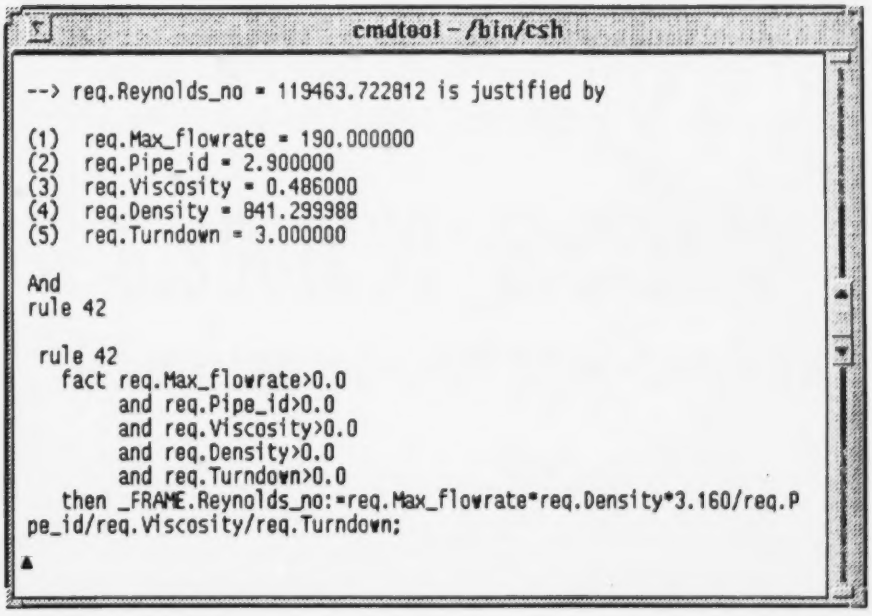

Figure 7.5 Example of explanation feature.

Although the user can determine the conclusion of the selection process by interpreting the displayed statements as described above, the knowledge bases of IDISDPFSS are developed so as to take advantage of Meta-COOP's graphic display capability. Thus, the final results are displayed in a graphic form. While the graphic form does not indicate why a specific flowmeter is selected or disqualified, as the statements can, the graphic representation does provide the conclusions in an easy to understand form. Figure 7.6 illustrates the conclusions as obtained using the requirements shown in Figure 7.2.

The "Fluid prop." button starts the viscosity coupling system running on the PC, while displaying the user interface on the Sun monitor and accepting input from the Sun keyboard. The coupling system has multiple screen displays of which only two are illustrated. Figure 7.7 shows a partial list of the components that can make-up the fluid mixture. Figure 7.8 shows the results predicted for a mixture containing methane, ethane and propane. The results indicate the viscosity method utilized, and the thermodynamic state, density, viscosity and compressibility factor of the fluid for both phases at the specified pressure and temperature.

When developing the IDISDPFSS interface, it was decided that the user would be allowed the option of invoking the coupling system rather than have IDISDPFSS display it automatically. There are three reasons for this. The first two reasons are because in many cases the user would already have the required fluid properties such as fluid state, density and viscosity, or would not have the complete composition of the fluid, as required by the coupling system. The third reason is that by providing a button to activate the coupling system and using the X Window System capabilities and network communications, the user can have more than one viscosity coupling system running concurrently as shown in Figure 7.9. This allows the user to run the coupling system with one set of data in one

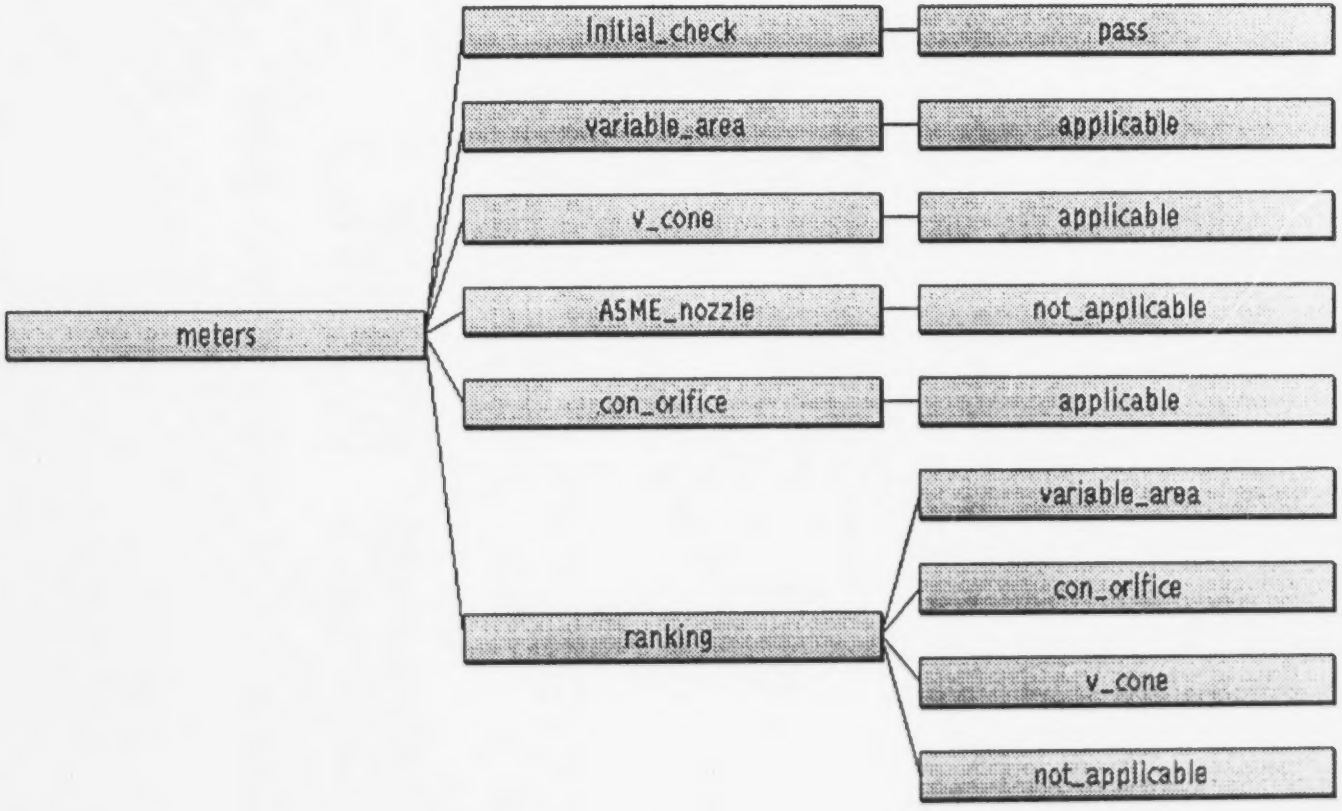

Figure 7.6 Graphic presentation of selection process conclusions.

Figure 7.7 Partial list of components available in the viscosity coupling system.

PC Plus
VISCOSITY COUPLING SYSTEM
*** Viscosity method - Pedersen & Fredenslund ***
*** Viscosity prediction at .7000D+02 F and .4000D+03 Psia ***
*** Component Composition, mole fraction K Factor ***
*** Feed Liquid Vapor ***
*** Methane .1000D+00 .9210D-01 .4298D+00 .4657D+01 ***
*** Ethane .2000D+00 .1989D+00 .2452D+00 .1233D+01 ***
*** Propane .7000D+00 .7088D+00 .3250D+00 .4585D+00 ***
*** ***
*** Density kg/m3 .4679D+03 .4181D+02 ***
*** Mole fraction .9772D+00 .2283D-01 ***
*** Vis cp .8001D-01 .1100D-01 ***
*** Z compressibility .9330D-01 .7712D+00 ***
** End - RETURN/ENTER to continue

Figure 7.8 Results from viscosity coupling system.

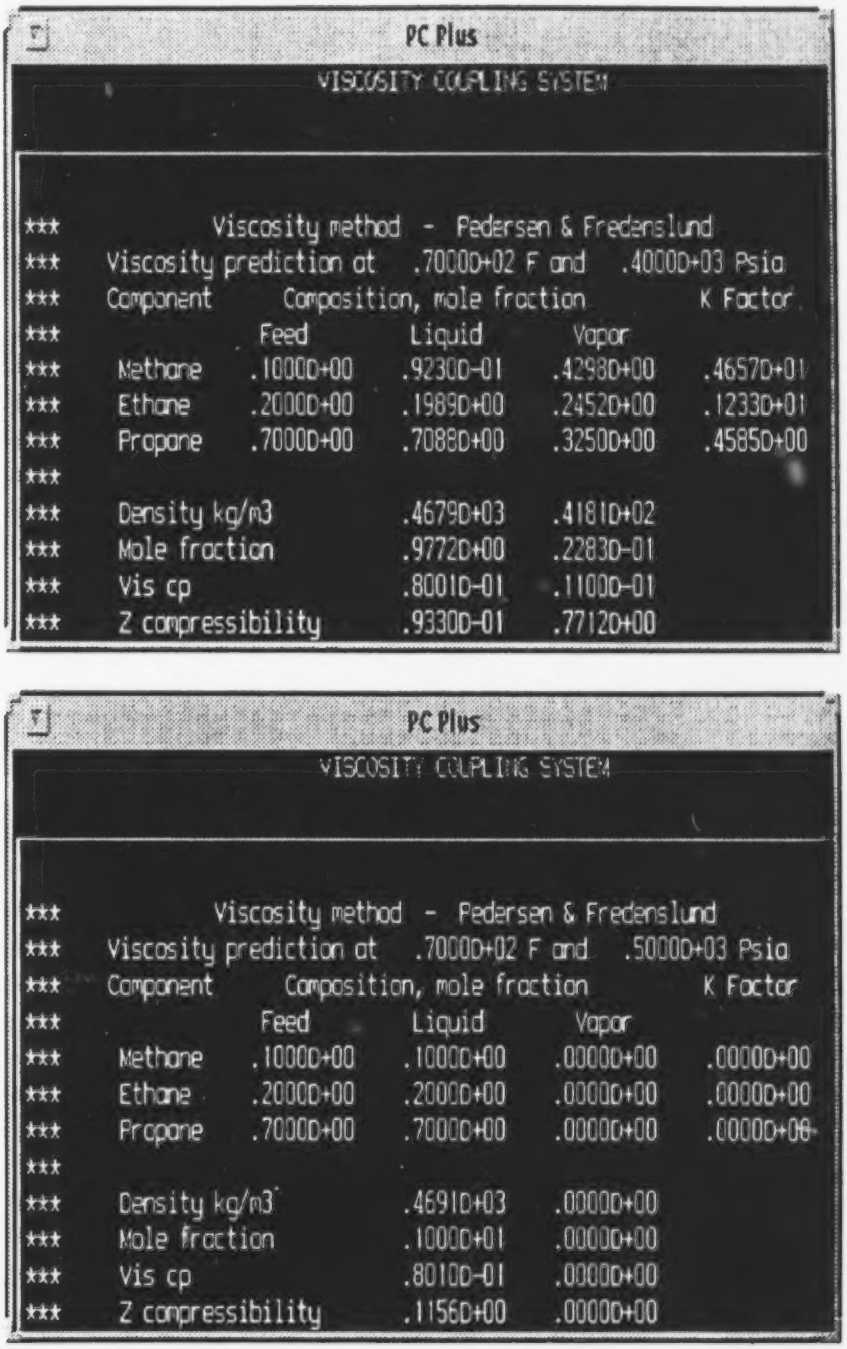
PC Plus
VISCOSITY COUPLING SYSTEM
*** Viscosity method - Pedersen & Fredenslund
*** Viscosity prediction at .7000D+02 F and .4000D+03 Psia
*** Component Composition, mole fraction K Factor
*** Feed Liquid Vapor
*** Methane .1000D+00 .9230D-01 .4298D+00 .4657D+01
*** Ethane .2000D+00 .1989D+00 .2452D+00 .1233D+01
*** Propane .7000D+00 .7088D+00 .3250D+00 .4585D+00
*** Density kg/m3 .4679D+03 .4181D+02
*** Mole fraction .9772D+00 .2283D-01
*** Vis cp .8001D-01 .1100D-01
*** Z compressibility .9330D-01 .7712D+00
PC Plus
VISCOSITY COUPLING SYSTEM
*** Viscosity method - Pedersen & Fredenslund
*** Viscosity prediction at .7000D+02 F and .5000D+03 Psia
*** Component Composition, mole fraction K Factor
*** Feed Liquid Vapor
*** Methane .1000D+00 .1000D+00 .0000D+00 .0000D+00
*** Ethane .2000D+00 .2000D+00 .0000D+00 .0000D+00
*** Propane .7000D+00 .7000D+00 .0000D+00 .0000D+00
*** Density kg/m3 .4691D+03 .0000D+00
*** Mole fraction .1000D+01 .0000D+00
*** Vis cp .8010D-01 .0000D+00
*** Z compressibility .1156D+00 .0000D+00

Figure 7.9 Illustration of concurrent usage of viscosity coupling system.

window and then start coupling system with another set of data in another window. Thus, the user can compare results displayed in the two windows.

The "Size" button, when activated by placing the pointer on the button and pressing the left mouse button, displays the pop-up menu shown in Figure 7.10 that contains four other buttons: "variable area", "V-Cone", "flow nozzle" and "orifice plate". These four buttons start the individual sizing software packages. Although these packages run on the PC, their interfaces are displayed on the Sun monitor and they receive data from the Sun keyboard. Figure 7.11† illustrates the screen in MEM 1.0 where the user selects equation 5.1, 5.2 or 5.3 to employ based on the fluid state and composition. Figure 7.12 presents the five display areas of V-Cone 3.1. Figure 7.13 shows one of the displays in the "Main Program Screens" in FLOWEL 2.0 where the user must enter specifications.

The decision to have the user activate the various flowmeter sizing packages is to take full advantage of the X Window System and network communications, and to provide maximum flexibility to the user. If after completing the selection process the IDISDPFSS automatically invokes the sizing package associated with the most appropriate flowmeter, this would limit the user to just one flowmeter type. However, by allowing the user to invoke the sizing packages independent from the selection process, the user may investigate the other flowmeter alternatives at his/her discretion. Also, the user can perform multiple sizings concurrently by using the same software package. To illustrate this feature, Figure 7.14 shows two FLOWEL 2.0 flowmeter sizings as displayed on the Sun monitor. Although this figure is difficult to read due to the resolution quality of the

---

† Unfortunately, the author has no control over the color scheme used in individual packages and only access to a monochrome printer. Thus, some figure may be difficult to read.

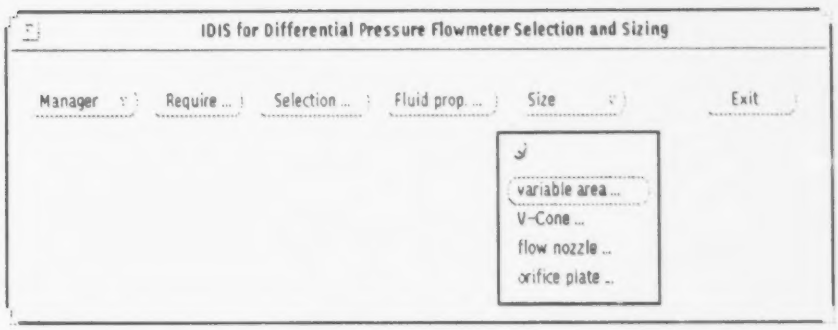

Figure 7.10 Pop-up menu for flowmeter sizing buttons

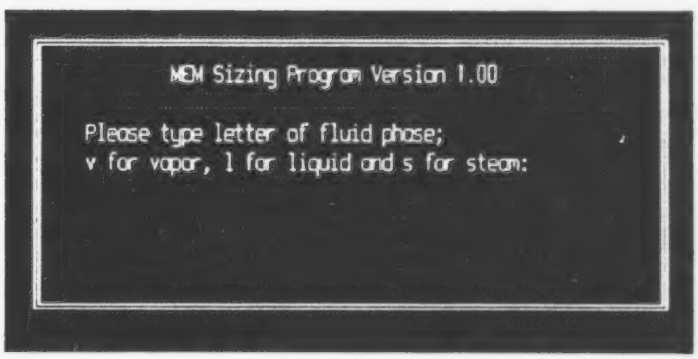

Figure 7.11 MEM 1 0 display window

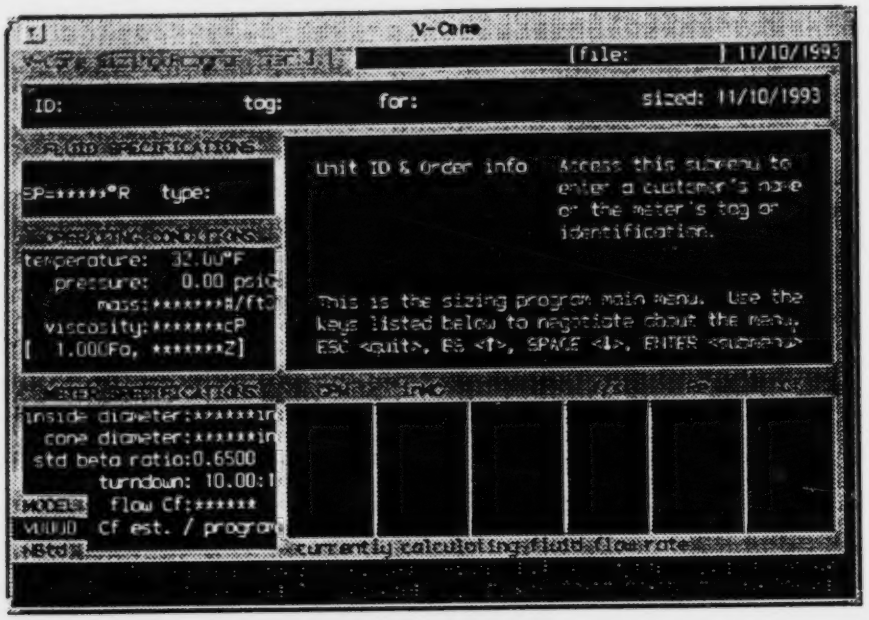
V-Cone
[file:] 11/10/1993
ID: tag: for: sized: 11/10/1993
Unit ID & Order info Access this submenu to enter a customer's name or the meter's tag or identification.
SP=*****°R type:
temperature: 32.00°F
pressure: 0.00 psig
mass:*******#/ft3
viscosity:*******cP
[1.000Fo, *******Z]
This is the sizing program main menu. Use the keys listed below to negotiate about the menu. ESC <quit>, BS <↑>, SPACE <↓>, ENTER <submenu>
inside diameter:*******in
cone diameter:*******in
std beta ratio:0.6500
turndown: 10.00:1
Flow Cf:******
Cf est. / program
currently calculating fluid flow rate

Figure 7.12 V-Cone 3.1 main screen.

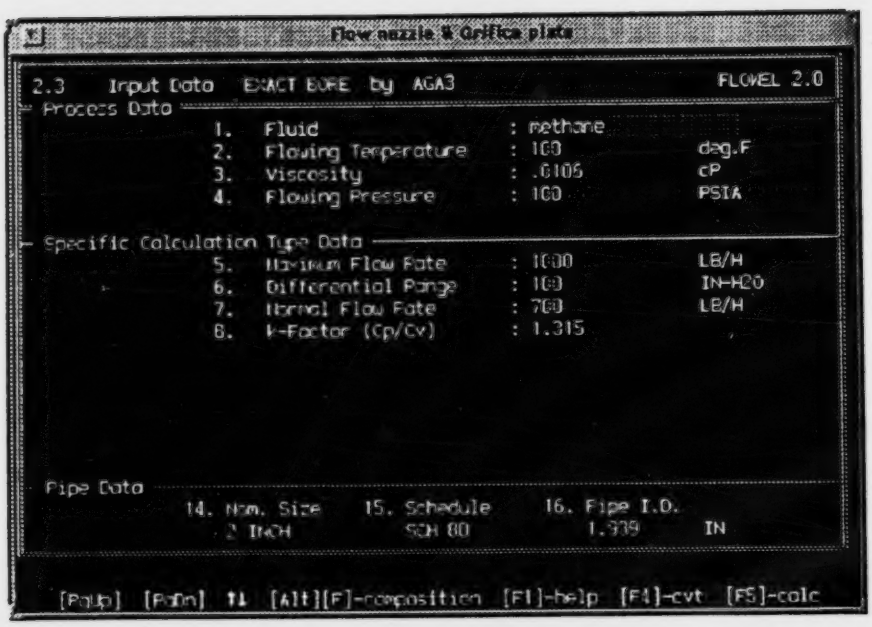
Flow nozzle & Orifice plate
2.3 Input Data EXACT BORE by AGA3 FLOWEL 2.0
Process Data
1. Fluid : methane
2. Flowing Temperature : 100 deg.F
3. Viscosity : .0105 cP
4. Flowing Pressure : 100 PSIA
Specific Calculation Type Data
5. Maximum Flow Rate : 1000 LB/H
6. Differential Range : 100 IN-H2O
7. Normal Flow Rate : 700 LB/H
8. K-Factor (Cp/Cv) : 1.315
Pipe Data
14. Nom. Size 15. Schedule 16. Pipe I.D.
2 INCH SCH 80 1.939 IN
[PgUp] [PgDn] ↑↓ [Alt][F]-composition [F1]-help [F4]-cvt [F5]-calc

Figure 7.13 FLOWEL 2.0 input screen.

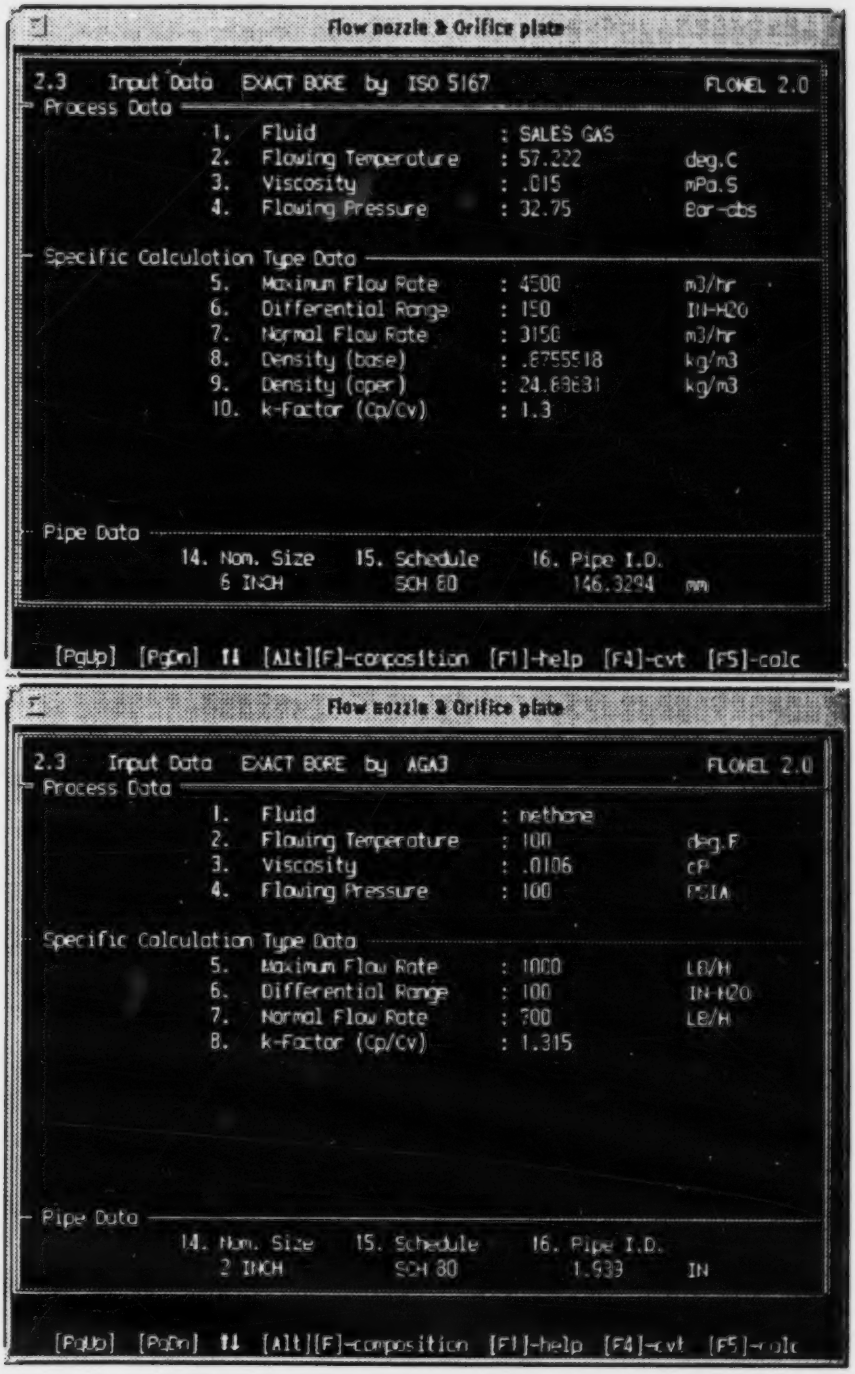
Flow nozzle & Orifice plate
2.3 Input Data EXACT BORE by ISO 5167 FLOWEL 2.0
Process Data
1. Fluid : SALES GAS
2. Flowing Temperature : 57.222 deg.C
3. Viscosity : .015 mPa.S
4. Flowing Pressure : 32.75 Bar-abs
Specific Calculation Type Data
5. Maximum Flow Rate : 4500 m3/hr
6. Differential Range : 150 IN-H2O
7. Normal Flow Rate : 3150 m3/hr
8. Density (base) : .8755518 kg/m3
9. Density (oper) : 24.68831 kg/m3
10. k-Factor (Cp/Cv) : 1.3
Pipe Data
14. Nom. Size 15. Schedule 16. Pipe I.D.
6 INCH SCH 80 146.3294 mm
[Pgup] [PgDn] [Alt][F]-composition [F1]-help [F4]-cvt [F5]-calc
Flow nozzle & Orifice plate
2.3 Input Data EXACT BORE by AGA3 FLOWEL 2.0
Process Data
1. Fluid : methane
2. Flowing Temperature : 100 deg.F
3. Viscosity : .0106 cP
4. Flowing Pressure : 100 PSIA
Specific Calculation Type Data
5. Maximum Flow Rate : 1000 LB/H
6. Differential Range : 100 IN-H2O
7. Normal Flow Rate : 700 LB/H
8. k-Factor (Cp/Cv) : 1.315
Pipe Data
14. Nom. Size 15. Schedule 16. Pipe I.D.
2 INCH SCH 80 1.939 IN
[Pgup] [PgDn] [Alt][F]-composition [F1]-help [F4]-cvt [F5]-calc

Figure 7.14 Illustration of concurrent usage of flowmeter sizing programs.

printer and size limitation of the printed page, the capability of concurrent sizing programs is demonstrated.

The reader might well wonder why not automatically display the sizing packages for all appropriate flowmeters selected. There are two reasons that this is not done. The first reason is that the monitor can become cluttered and difficult to read if all four types of flowmeters are appropriate. Figure 7.14 emphasizes this problem, although the monitor is much clearer than shown in the figure. The second reason is that by allowing the user to activate the sizing packages manually, the intelligent selection can be bypassed. Thus, a system that only provides concurrent sizing of different flowmeter types is available in one environment.

Chapter 8

Conclusions

For this thesis, three software packages have been developed: MEM 1.0, viscosity coupling system and integrated distributed intelligent system for differential pressure flowmeter selection and sizing (IDISDPFSS).

MEM 1.0 is developed to size the variable area flowmeters manufactured by Meter Equipment Manufacturing, Inc. This software package can size flowmeters for three fluid types: liquid, gas and steam. The software is written in the language C and compiled to run on a personal computer with DOS.

The viscosity coupling system is a software package to assist non-expert users in selecting an appropriate viscosity model to predict the viscosity for the users' defined fluids. The package contains three viscosity models: Dean and Stiel, Ely and Hanley, as well as Pedersen and Fredenslund. The knowledge concerning the models is captured with the LISP based commercial AI expert system tool PC Plus. In order to predict the viscosity, the knowledge in PC Plus is coupled to a FORTRAN program. This FORTRAN program not only predicts the viscosity but also the fluid state, density and compressibility factor. These fluid properties are determined using the Peng-Robinson equation of state. The coupling system runs on a personal computer with DOS.

IDISDPFSS is an integrated coordinated knowledge environment to assist non-expert users select and size differential pressure flowmeters for users' specific applications. There are four differential flowmeter types: MEM variable area, V-Cone, flow nozzle and orifice plate. These flowmeters are sized with the corresponding in-house software MEM 1.0 and the commercial software packages V-Cone 3.1 and FLOWEL 2.0, used for sizing both flow nozzles and orifice plates. The viscosity coupling system provides information required in the selection and sizing process.

The knowledge environment is based on the concept of an integrated distributed intelligent system of which the key construct is the meta-system. In order to implement the meta-system, Meta-COOP 2.0 is employed. Meta-COOP 2.0 utilizes units to capture expert knowledge and reasoning in the form of production rules and object-oriented programming.

IDISDPFSS has been developed in such a way that knowledge can be easily added in the future. This is accomplished by having one controlling unit that sends messages to other units. These other units provide a specific function such as determining whether a particular flowmeter type is applicable based on nineteen user's specified requirements. By decomposing this functionality into separate units, the knowledge relevant to a specific flowmeter type is contained in a single unit. Thus, if additional knowledge needs to be added, only one unit must be altered. In addition, if a new flowmeter type is to be added, only the unit containing the pertinent knowledge needs to be assembled. The other units do not require alteration.

IDISDPFSS provides a user-friendly graphic interface. The graphics allow information to be presented both alpha-numerically and graphically. The interface is a mouse-driven

window menu. The mouse allows the user to simply click a button to perform activities. The window environment permits several windows to be run concurrently. Since in most cases the different parts of the menu are independent of the other parts, the user has the flexibility to activate the different functions at his or her discretion.

IDISDPFSS is distributed such that the in-house and commercial software packages are autonomous and capable of functioning independently. The in-house software included MEM 1.0, Meta-COOP 2.0 and a viscosity coupling system. The commercial packages included FLOWEL 2.0, PC Plus and V-Cone 3.1.

IDISDPFSS is so integrated that it combines several individual software packages into a coordinated knowledge environment. The integration involves symbolic reasoning and numerical computations; numerous computer languages such as C, FORTRAN and LISP; in-house developed and commercial software packages (Meta-COOP vs. PC Plus, MEM 1.0 vs. FLOWEL 2.0 and V-Cone 3.1); heterogenous platforms (workstation and personal computer); and heterogenous operating systems (UNIX and DOS). The key to the integration is the use of network communications between the UNIX workstation and the DOS personal computer, and the use of the X Window System on both computers. This setup enables all programs whether running on the workstation or personal computer to be displayed and receive input from the workstation.

References

Anon. (1993). *DESQview/X User Guide*. Quarterdeck Office Systems. 500US-DX0100. Ventura Professional.

Anon. (1990). *System and Network Adminstration*. Sun Microsystems. Part number: 800-3805-10. (rev. A)

Anon. (1987). *Personal Consultant Plus Getting Started*. Part number: 2232456-0001. Austin: Publishing Center, Texas Instruments Inc., Data System Group.

Baker-Counsell, J. (1985). Flowmeter selection: Expert help is on its way. *Process Engineering*, *66*(10), 71,73.

Buchanan, B.G. (1985). Expert systems. *Journal of Automatic Reasoning*, *1*(1), 28-34.

Cheremisinoff, N.P. (1979). *Applied fluid measurement: Fundamentals and technology*. New York: Marcel Dekker.

Corbin, J. (1992). *Intelligent operation support system for a batch sulphite pulping process*. M.Sc. thesis. University of Alberta. Edmonton, Canada.

C-WIN (1993). *C-WIN Version 1.0 public domain software*. B. Withers, 649 Meadowbrook Street, Allen, Texas.

Danen, G.W.A. (Ed.). (1985). *Shell Flowmeter Engineering Handbook* (2nd ed.). New York: McGraw-Hill Book Company.

Dean, D.E., and Stiel, L.I. (1965). The viscosity of nonpolar gas mixtures at moderate and high pressures. *AIChE Journal*, *11*, 526-32.

Ely, J.F., and Hanley, H.J.M. (1981). Prediction of transport properties. 1. Viscosity of fluids and mixtures. *Ind & Engng Chem Fundam*, *20*(4), 323-31.

Fikes, R., and Kehler, T. (1985). The role of frame-based representation in reasoning. *Communications of the ACM*, *28*(9), 904-20.

Gani, R., and O'Connell, J.P. (1989). A knowledge based system for the selection of thermodynamic models. *Computers & Chem Engng, 13*(4/5), 397-404.

Ginesi, D. (1991). Choosing the best flowmeter. *Chemical Engineering, 98*(4), 88-100.

Ginesi, D., and Grebe, G. (1985). Flowmeter selection: A comparison of performance features vs. economic costs. *Advances in Instrumentation and Control: Proceedings of the 1985 ISA International Conference and Exhibit, 40*, pt. 2, 1173-90.

Hasley, D.M. (1986). Survey of industrial usage of flowmeters. *Measurement and Control, 19*(5), 52-55.

Hayward, A.T. (1979). *Flowmeters: A Basic Guide and Source-Book for Users*. New York: John Wiley & Sons.

Jossi, J.A., Stiel, L.I., and Thodos, G. (1962). The viscosity of pure substances in the dense gaseous and liquid phases. *AIChE Journal, 8*(1), 59-62.

Kitzmiller, C.T., and Kowalik, J.S. (1987). Coupling symbolic and numeric computing in knowledge-based systems. *AI Magazine, 8*(2), 85-90.

Lipták, B.G., and Venczel, K. (Eds.). (1982). *Instrument Engineers' Handbook* (rev ed.). Pennsylvania: Chilton Book Company.

Lomas, D.J. (1986). Selecting flowmeters for industrial applications. *Brown Boveri Review, 73*(2), 69-79.

Lomas, D.J. (1977). Selecting the right flowmeter part 1: The six favorites. *Instrumentation Technology, 24*(5), 55-62.

Lycett, J., and Maudsley, D. (1986). Development of an expert system for flowmeter selection. *Measurement and Control, 19*(9), 251-52.

Meinhold, T.F. (1984). Liquid flowmeters. *Plant Engineering*, 38(28), 46-60.

Miller, R.W. (1983). *Flow Measurement Engineering Handbook*. New York: McGraw-Hill Book Company.

O'Brien, C. (1989). Flowmeter terms, types & successful selection. *InTech, 36*(12), 30-33.

Opie, R. (1987). Getting the most out of flowmetering. *Control and Instrumentation, 19*(2), 31-32.

Pedersen, K.S., and Fredenslund, A. (1987). An improved corresponding states model for the prediction of oil and gas viscosity and thermal. *Chem Eng Sci, 40*(1), 182-87.

Peng, D, and Vermani, R. (1987). An empirical method for calculating the viscosity of hydrocarbon liquid and liquid mixtures. *AIChE Spring National Meeting*, Mar. 29 - Apr. 2, Houston, Texas.

Peng, D.-Y., and Robinson, D.B. (1976). A new two-constant equation of state. *Ind. Eng. Chem. Fundam., 15*, 59-64.

Radcliffe, M. (1992). A technical perspective for the '90s. Byte, *17*(12), 144 QD-1 -QD-30.

Rao, M., Wang, Q., and Cha, J. (1993). *Integrated Distributed Intelligent Systems in Manufacturing*. London: Chapman & Hall.

Rao, M., and Qiu, H. (1993). *Process Control Engineering*. Langhorne: Gordon and Breach Science Publishers, Inc.

Rao, M. (1991). *Integrated System for Intelligent Control*. Berlin: Springer-Verlag.

Rao, M. (1992). Frontiers and challenges of intelligent process control. *Engineering Applications of Artificial Intelligence, 5*(6), 475-81.

Rao, M., Jiang, T., and Tsai, J. (1989). Combining symbolic and numerical processing for real-time intelligent control. *Engineering Applications of Artificial Intelligence, 2*(3), 19-27.

Rao, M., Tsai, J., and Jiang, T. (1988). An intelligent decisionmaker for optimal control. *Applied Artificial Intelligence, 2*, 285-305.

Rao, M., Tsai, J., and Jiang, T.S. (1987). A framework of integrated intelligent systems. *Proc. IEEE Intern. Conf. on System, Man and Cybernetics*, Alexandria, Virginia, 1133-37.

Rusnak, J. (1989). Fundamentals of flowmeter selection. *InTech*, *36*(4), 49-51.

Rutz, P. (1991). An introduction to object-oriented programming. *CACHE News*, Fall, 3-7.

Sovik, R.E. (1985). Flow measurement - Some new considerations. *Mechanical Engineering*, *107*(5), 48-52.

Spitzer, D.W. (1985). The effects of fluid properties on flowmeter performance: A user's perspective. *Advances in Instrumentation and Control: Proceedings of the 1985 ISA International Conference and Exhibit*, *40*, pt. 2, 1167-71.

Stefik, M., and Bobrow, D. (1986). Object-oriented programming: Themes and variations. *AI Magazine*, *6*(4), 40-62.

Ten Dyke, R.P., and Kunz, J.C. (1989). Object-oriented programming. *IBM Systems Journal*, *28*(3), 465-78.

Tham, M.J., and Gubbins, K.E. (1970). Correspondence principle for transport properties of dense fluids. *Ind & Engng Chem Fundam*, *9*, 63-70.

Wiener, R., and Sincovec, R. (1984). *Software Engineering with Modula-2 and Ada*. New York: Wiley & Sons.

Wilcox, J. (1988). *FLOWEL 2.0 Flow Element Sizing and Documentation*. Version 2.0, 88-11-21. Xerox Ventura Publishing.

Wong, F.S., Dong, W., and Blanks, M. (1988). Coupling of symbolic and numerical computations on a microcomputer. *Artificial Intelligence in Engineering*, *3*(1), 32-38.

Appendix A

IDISDPFSS Knowledge Bases

This appendix contains the source code for the eight knowledge bases in IDISDPFSS: flow.kbs, flow_req.kbs, flow_check.kbs, flow_var_area.kbs, flow_vcone.kbs, flow_nozzle.kbs, flow_orifice.kbs and flow_eval.kbs.

```
/*****************************************************************************/
/*      Module          : flow.kbs                                           */
/*      Function        : Root frame of flowmeter selection                  */
/*                                                                           */
/*      Created by      : Murray Stevenson                                   */
/*                                                                           */
/*****************************************************************************/
GLOBE
 meters : flowmeter;
END


Unit: flowmeter in_knowledge_base flow.kbs;

Memberslot: initial_check from flowmeter;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: initial_check;
  Values: Unknown;
End Slot;

Memberslot: variable_area from flowmeter;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: variable_area;
  Values: Unknown;
End Slot;

Memberslot: vcone from flowmeter;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: v_cone;
  Values: Unknown;
End Slot;

Memberslot: ASME_nozzle from flowmeter;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: ASME_nozzle;
  Values: Unknown;
End Slot;

Memberslot: orifice_plate from flowmeter;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: con_orifice;
  Values: Unknown;
End Slot;

Memberslot: evaluation from flowmeter;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: evaluation;
  Values: Unknown;
End Slot;

Memberslot: choose_method from flowmeter;
  Inheritance: METHOD;
  Valueclass: METHODS;
```

```
  Values: choose_method;
End Slot;

End Unit;

METHOD choose_method(choose:keyword)
VAR
 x:integer;
BEGIN

 send ("input flow_data") to "req";
 _FRAME.initial_check:="initial_check";
 send ("check") to "initial_check";
 if initial_check.check<>"fail" then
 begin
   _FRAME.variable_area:="variable_area";
   send ("check") to "variable_area";
   _FRAME.vcone:="v_cone";
   send ("check") to "v_cone";
   _FRAME.ASME_nozzle:="ASME_nozzle";
   send ("check") to "ASME_nozzle";
  _FRAME.orifice_plate:="con_orifice";
  send ("check") to "con_orifice";
  _FRAME.evaluation:="ranking";
  send ("eval") to "ranking";
 end;
END.
```

```
/*****************************************************************************/
/*      Module          : flow_req.kbs                                       */
/*      Function        : Root frame of flowmeter selection                  */
/*                                                                           */
/*      Created by      : Murray Stevenson                                   */
/*                                                                           */
/*****************************************************************************/
GLOBE
 req : user_input;
END


Unit: user_input in_knowledge_base flow_require.kbs;

Memberslot: Application from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: string;
  Values: Unknown;
End Slot;

Memberslot: Primary_purpose from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: string;
  Values: Unknown;
End Slot;

Memberslot: Max_press from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: real;
  Values: Unknown;
End Slot;

Memberslot: Max_temp from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: real;
  Values: Unknown;
End Slot;

Memberslot: Particulate_matter from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: string;
  Values: Unknown;
End Slot;

Memberslot: Pipe_size from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: real;
  Values: Unknown;
End Slot;

Memberslot: Vertical_mount from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: string;
```

```
  Values: Unknown;
End Slot;

Memberslot: Remote_readout from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: string;
  Values: Unknown;
End Slot;

Memberslot: Max_flowrate from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: real;
  Values: Unknown;
End Slot;

Memberslot: Pipe_id from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: real;
  Values: Unknown;
End Slot;

Memberslot: Viscosity from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: real;
  Values: Unknown;
End Slot;

Memberslot: Density from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: real;
  Values: Unknown;
End Slot;

Memberslot: Reynolds_no from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: real;
  Values: Unknown;
End Slot;

Memberslot: Accuracy from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: real;
  Values: Unknown;
End Slot;

Memberslot: Reverse_flow from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: string;
  Values: Unknown;
End Slot;

Memberslot: Upstream_pipe_run from user_input;
  Inheritance: OVERRIDE.VALUES;
```

```
  Valueclass: integer;
  Values: Unknown;
End Slot;

Memberslot: Downstream_pipe_run from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: integer;
  Values: Unknown;
End Slot;

Memberslot: Turndown from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: real;
  Values: Unknown;
End Slot;

Memberslot: Ero_corrosive_fluid from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: string;
  Values: Unknown;
End Slot;

Memberslot: Max_diff_press from user_input;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: real;
  Values: Unknown;
End Slot;

Memberslot: Rey_rules from user_input;
  Inheritance: Override.Values;
  Valueclass: RULES ;
  Values: {
        rule 42
          fact req.Max_flowrate>0.0
              and req.Pipe_id>0.0
              and req.Viscosity>0.0
              and req.Density>0.0
              and req.Turndown>0.0
          then
_FRAME.Reynolds_no:=req.Max_flowrate*req.Density*3.160/req.Pipe_id/req.Viscosity/req.Turndown;
        }
End Slot;


Memberslot: input_require from user_input;
   Inheritance:METHOD;
   Valueclass:METHODS;
   Values:input_require;
End Slot;

End UNIT;

METHOD input_require(input:keyword filename:string)
```

```
VAR
  x:real;

BEGIN
/*  call extern function inputrequire() input design
    requirement data
*/
 inputrequire(filename);
 reason(_FRAME,"Rey_rules");
END.
```

```
/*****************************************************************************/
/*      Module          : flow_check.kbs                                     */
/*      Function        : Initial check of all flowmeter types               */
/*                                                                           */
/*      Created by      : Murray Stevenson                                   */
/*                                                                           */
/*****************************************************************************/
GLOBE
 initial_check : initial_meters_check;
END

Unit: initial_meters_check in_knowledge_base flow_check.kbs;


Memberslot: check  from initial_meters_check;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: check;
  Values: Unknown;
End Slot;

Memberslot: check_default  from initial_meters_check;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: string;
  Values: "okay";
End Slot;

Memberslot: check_rules from initial_meters_check;
  Inheritance: Override.Values;
  Valueclass: RULES ;
  Values: {
        rule 1
         fact req.Max_press>=6000.0
         then _FRAME.check="fail";
             _FRAME.check_default="fail"

        rule 2
         fact req.Max_temp>=93.33
            and req.Max_press>=5830.0
         then _FRAME.check="fail";
             _FRAME.check_default="fail"

        rule 3
         fact req.Max_temp>=148.89
            and req.Max_press>=5690.0
         then _FRAME.check="fail";
             _FRAME.check_default="fail"

        rule 4
         fact req.Max_temp>=204.44
            and req.Max_press>=5550.0
         then _FRAME.check="fail";
             _FRAME.check_default="fail"
```

```
rule 5
 fact req.Max_temp>=260.00
    and req.Max_press>=5210.0
 then _FRAME.check="fail";
    _FRAME.check_default="fail"

rule 6
 fact req.Max_temp>=315.56
    and req.Max_press>=4620.0
 then _FRAME.check="fail";
    _FRAME.check_default="fail"

rule 7
 fact req.Max_temp>=371.11
    and req.Max_press>=3920.0
 then _FRAME.check="fail";
    _FRAME.check_default="fail"

rule 8
 fact req.Max_temp>=426.67
    and req.Max_press>=3050.0
 then _FRAME.check="fail";
    _FRAME.check_default="fail"

rule 9
 fact req.Max_temp>=454.44
    and req.Max_press>=2500.0
 then _FRAME.check="fail";
    _FRAME.check_default="fail"

rule 10
 fact req.Pipe_size<0.5
 then _FRAME.check="fail";
    _FRAME.check_default="fail"

rule 11
 fact req.Accuracy<0.75
 then _FRAME.check="fail";
    _FRAME.check_default="fail"

rule 12
 fact req.Reverse_flow="yes"
 then _FRAME.check="fail";
    _FRAME.check_default="fail"

rule 13
 fact req.Turndown>10.0
 then _FRAME.check="fail";
    _FRAME.check_default="fail"

rule 14
 fact req.Max_diff_press<5.0
 then _FRAME.check="fail";
```

```
            _FRAME.check_default="fail"

          rule 15
            fact _FRAME.check_default="okay"
            then _FRAME.check="pass"
          }

End Slot;

Memberslot: check_method from initial_meters_check;
  Inheritance: METHOD;
  Valueclass: METHODS;
  Values: check_method;
End Slot;

End Unit;

METHOD check_method(check:keyword)
VAR
  x : string;

BEGIN
 reason(_FRAME,"check_rules");
END.
```

```
/*****************************************************************************/
/*      Module      : flow_var_area.kbs                                      */
/*      Function    : Check variable area flowmeter for application          */
/*                                                                           */
/*      Created by  : Murray Stevenson                                       */
/*                                                                           */
/*****************************************************************************/
GLOBE
 variable_area : var_area_check;
END

Unit: var_area_check in_knowledge_base flow_var_area.kbs;


Memberslot: check  from var_area_check;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: check;
  Values: Unknown;
End Slot;

Memberslot: check_default  from var_area_check;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: string;
  Values: "okay";
End Slot;

Memberslot: Centistokes  from var_area_check;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: real;
  Values: Unknown;
End Slot;

Memberslot: check_rules from var_area_check;
  Inheritance: Override.Values;
  Valueclass: RULES ;
  Values: {
        rule 21
          fact req.Max_press>=1000.0
          then _FRAME.check="not_applicable";
              _FRAME.check_default="fail"

        rule 22
          fact req.Max_temp>=315.56
          then _FRAME.check="not_applicable";
              _FRAME.check_default="fail"

        rule 23
          fact req.Pipe_size>4.0
          then _FRAME.check="not_applicable";
              _FRAME.check_default="fail"

        rule 24
          fact req.Max_temp>=204.44
```

```
    and req.Pipe_size>1.5
 then _FRAME.check="not_applicable";
    _FRAME.check_default="fail"

rule 25
 fact req.Pipe_size>1.5
    and req.Max_press>=440.0
 then _FRAME.check="not_applicable";
    _FRAME.check_default="fail"

rule 26
 fact req.Max_temp>=60.0
    and req.Pipe_size<=4.0
 then _FRAME.check="not_applicable";
    _FRAME.check_default="fail"

rule 27
 fact req.Pipe_size<=4.0
    and req.Max_press>=300.0
 then _FRAME.check="not_applicable";
    _FRAME.check_default="fail"

rule 28
 fact req.Particulate_matter="yes"
 then _FRAME.check="not_applicable";
    _FRAME.check_default="fail"

rule 29
 fact req.Vertical_mount="no"
 then _FRAME.check="not_applicable";
    _FRAME.check_default="fail"

rule 30
 fact req.Remote_readout="yes"
 then _FRAME.check="not_applicable";
    _FRAME.check_default="fail"

rule 31
 fact req.Pipe_size<=0.75
 then _FRAME.check="not_applicable";
    _FRAME.check_default="fail"

rule 32
 fact req.Primary_purpose="control"
 then _FRAME.check="not_applicable";
    _FRAME.check_default="fail"

rule 33
 fact req.Primary_purpose="accounting"
 then _FRAME.check="not_applicable";
    _FRAME.check_default="fail"

rule 34
```

```
        fact req.Accuracy<2.0
        then _FRAME.check="not_applicable";
            _FRAME.check_default="fail"

      rule 35
        fact _FRAME.Centistokes>5.0
        then _FRAME.check="not_applicable";
            _FRAME.check_default="fail"

      rule 36
        fact _FRAME.check_default="okay"
        then _FRAME.check="applicable"
      }

End Slot;

Memberslot: check_method1 from var_area_check;
  Inheritance: METHOD;
  Valueclass: METHODS;
  Values: check_method1;
End Slot;

End Unit;

METHOD check_method1(check.keyword)
VAR
  x : string;

BEGIN
  _FRAME.Centistokes:=req.Viscosity*1000.0/req.Density;
  reason(_FRAME,"check_rules");
END.
```

```
/*****************************************************************************/
/*      Module      : flow_vcone.kbs                                         */
/*      Function    : Check v-cone flowmeter for application                 */
/*                                                                           */
/*      Created by  : Murray Stevenson                                       */
/*                                                                           */
/*****************************************************************************/
GLOBE
 v_cone : vcone_check;
END

Unit: vcone_check in_knowledge_base flow_vcone kbs;


Memberslot: check from vcone_check;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: check;
  Values: Unknown;
End Slot;

Memberslot: check_default from vcone_check;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: string;
  Values: "okay";
End Slot;

Memberslot: velocity from vcone_check;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: real;
  Values: Unknown;
End Slot;

Memberslot: check_rules from vcone_check;
  Inheritance: Override.Values;
  Valueclass: RULES ;
  Values: {
        rule 50
         fact req.Application="liquid"
            and _FRAME.velocity>=30.0
         then _FRAME.check="not_applicable";
            _FRAME.check_default="fail"

        rule 51
         fact req.Application="gas"
            and _FRAME.velocity>=800.0
         then _FRAME.check="not_applicable";
            _FRAME.check_default="fail"

        rule 52
         fact req.Reynolds_no<=8000.0
         then _FRAME.check="not_applicable";
            _FRAME.check_default="fail"
```

```
        rule 53
          fact req.Upstream_pipe_run<2
          then _FRAME.check="not_applicable";
              _FRAME.check_default="fail"

        rule 54
          fact req.Downstream_pipe_run<5
          then _FRAME.check="not_applicable";
              _FRAME.check_default="fail"

        rule 55
          fact _FRAME.check_default="okay"
          then _FRAME.check="applicable"
        }

End Slot;

Memberslot: check_method2 from vcone_check;
  Inheritance: METHOD;
  Valueclass: METHODS;
  Values: check_method2;
End Slot;

End Unit;

METHOD check_method2(check.keyword)
VAR
  x : string;

BEGIN
  _FRAME.velocity:=req.Max_flowrate*0.40874/req.Pipe_id/req.Pipe_id;
  reason(_FRAME,"check_rules");
END.
```

```
/*****************************************************************************/
/*      Module      : flow_nozzle.kbs                                        */
/*      Function    : Check flow nozzle flowmeter for application            */
/*                                                                           */
/*      Created by  : Murray Stevenson                                       */
/*                                                                           */
/*****************************************************************************/
GLOBE
 ASME_nozzle : nozzle_check;
END

Unit: nozzle_check in_knowledge_base flow_nozzle.kbs;


Memberslot: check  from nozzle_check;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: check;
  Values: Unknown;
End Slot;

Memberslot: check_default  from nozzle_check;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: string;
  Values: "okay";
End Slot;

Memberslot: check_rules from nozzle_check;
  Inheritance: Override.Values;
  Valueclass: RULES ;
  Values: {
        rule 60
          fact req.Pipe_size<6.0
          then _FRAME.check="not_applicable";
              _FRAME.check_default="fail"

        rule 61
          fact req.Reynolds_no<=200000.0
          then _FRAME.check="not_applicable";
              _FRAME.check_default="fail"

        rule 62
          fact req.Accuracy<2.0
          then _FRAME.check="not_applicable";
              _FRAME.check_default="fail"

        rule 63
          fact req.Turndown>3.0
          then _FRAME.check="not_applicable";
              _FRAME.check_default="fail"

        rule 64
          fact req.Upstream_pipe_run<10
          then _FRAME.check="not_applicable";
```

```
            _FRAME.check_default="fail"

          rule 65
            fact req.Downstream_pipe_run<5
            then _FRAME.check="not_applicable";
               _FRAME.check_default="fail"

          rule 66
            fact req.Max_diff_press<50.0
            then _FRAME.check="not_applicable";
               _FRAME.check_default="fail"

          rule 67
            fact _FRAME.check_default="okay"
            then _FRAME.check="applicable"
          }

End Slot;

Memberslot: check_method3 from nozzle_check;
  Inheritance: METHOD;
  Valueclass: METHODS;
  Values: check_method3;
End Slot;

End Unit;

METHOD check_method3(check keyword)
VAR
  x : string;

BEGIN
 reason(_FRAME,"check_rules");
END.
```

```
/*****************************************************************************/
/*      Module      : flow_orifice.kbs                                       */
/*      Function    : Check concentric orifice flowmeter for appl.           */
/*                                                                           */
/*      Created by  : Murray Stevenson                                       */
/*                                                                           */
/*****************************************************************************/
GLOBE
 con_orifice : orifice_check;
END

Unit: orifice_check in_knowledge_base flow_orifice.kbs;


Memberslot: check  from orifice_check;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: check;
  Values: Unknown;
End Slot;

Memberslot: check_default  from orifice_check;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: string;
  Values: "okay";
End Slot;

Memberslot: check_rules from orifice_check;
  Inheritance: Override.Values;
  Valueclass: RULES ;
  Values: {
        rule 70
          fact req.Pipe_size<2.0
          then _FRAME.check="not_applicable";
              _FRAME.check_default="fail"

        rule 71
          fact req.Reynolds_no<=10000.0
          then _FRAME.check="not_applicable";
              _FRAME.check_default="fail"

        rule 72
          fact req.Particulate_matter="yes"
          then _FRAME.check="not_applicable";
              _FRAME.check_default="fail"

        rule 73
          fact req.Turndown>3.0
          then _FRAME.check="not_applicable";
              _FRAME.check_default="fail"

        rule 74
          fact req.Upstream_pipe_run<10
          then _FRAME.check="not_applicable";
```

```
            _FRAME.check_default="fail"

          rule 75
           fact req.Downstream_pipe_run<5
           then _FRAME.check="not_applicable";
              _FRAME.check_default="fail"

          rule 76
           fact req.Ero_corrosive_fluid="yes"
           then _FRAME.check="not_applicable";
              _FRAME.check_default="fail"

          rule 77
           fact req.Max_diff_press<50.0
           then _FRAME.check="not_applicable";
              _FRAME.check_default="fail"

          rule 78
           fact _FRAME.check_default="okay"
           then _FRAME.check="applicable"
          }

End Slot;

Memberslot: check_method4 from orifice_check;
  Inheritance: METHOD;
  Valueclass: METHODS;
  Values: check_method4;
End Slot;

End Unit;

METHOD check_method4(check:keyword)
VAR
  x : string;

BEGIN
 reason(_FRAME,"check_rules");
END.
```

```
/*****************************************************************************/
/*      Module      : flow_eval.kbs                                          */
/*      Function    : Check concentric orifice flowmeter for appl.           */
/*                                                                           */
/*      Created by  : Murray Stevenson                                       */
/*                                                                           */
/*****************************************************************************/
GLOBE
 ranking : evaluation;
END

Unit: evaluation in_knowledge_base flow_eval.kbs;


Memberslot: rank1  from evaluation;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: rank1;
  Values: "NULL";
End Slot;

Memberslot: rank2  from evaluation;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: rank2;
  Values: "NULL";
End Slot;

Memberslot: rank3  from evaluation;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: rank3;
  Values: "NULL";
End Slot;

Memberslot: rank4  from evaluation;
  Inheritance: OVERRIDE.VALUES;
  Valueclass: rank4;
  Values: "NULL";
End Slot;

Memberslot: eval_method from evaluation;
  Inheritance: METHOD;
  Valueclass: METHODS;
  Values: eval_method;
End Slot;

End Unit;

METHOD eval_method(eval:keyword)
VAR
  x : string;

BEGIN

 if (_FRAME.rank1="NULL") and (variable_area.check="applicable") then
```

```
begin
  _FRAME.rank1:="variable_area";
end;
if (_FRAME.rank1="NULL") and (con_orifice.check="applicable") then
begin
  _FRAME.rank1:="con_orifice";
  _FRAME.rank4:="not_applicable";
end;
if (_FRAME.rank1="NULL") and (ASME_nozzle.check="applicable") then
begin
  _FRAME.rank1:="ASME_nozzle";
  _FRAME.rank3:="not_applicable";
  _FRAME.rank4:="not_applicable";
end;
if (_FRAME.rank1="NULL") and (v_cone.check="applicable") then
begin
  _FRAME.rank1:="v_cone";
  _FRAME.rank2:="not_applicable";
  _FRAME.rank3:="not_applicable";
  _FRAME.rank4:="not_applicable";
end;
if (_FRAME.rank2="NULL") and (con_orifice.check="applicable") then
begin
  _FRAME.rank2:="con_orifice";
end;
if (_FRAME.rank2="NULL") and (ASME_nozzle.check="applicable") then
begin
  _FRAME.rank2:="ASME_nozzle";
  _FRAME.rank4:="not_applicable";
end;
if (_FRAME.rank2="NULL") and (v_cone.check="applicable") then
begin
  _FRAME.rank1:="v_cone";
  _FRAME.rank3:="not_applicable";
  _FRAME.rank4:="not_applicable";
end;
if (_FRAME.rank3="NULL") and (ASME_nozzle.check="applicable") then
begin
  _FRAME.rank3:="ASME_nozzle";
end;
if (_FRAME.rank3="NULL") and (v_cone.check="applicable") then
begin
  _FRAME.rank3:="v_cone";
  _FRAME.rank4:="not_applicable";
end;
if (_FRAME.rank4="NULL") and (v_cone.check="applicable") then
begin
  _FRAME.rank4:="v_cone";
end;
END.
```

Appendix B

MEM 1.0 Source Code

This appendix contains the source code for the main program of MEM 1.0 for sizing Meter Equipment Manufacturing, Inc. variable area flowmeters. The program has multi-window user interface that is controlled using the keyboard. The program is written in C.

```
#include <stdlib.h>
#include <stddef.h>
#include <time.h>
#include <conio.h>
#include <bios.h>
#include <ctype.h>
#include <string.h>
#include <math.h>
#include "win.h"
#include "keys.h"

int phase;
float f[10];

static void Delay(nSeconds)
short       nSeconds;
{
   auto    time_t   lQuitTime;

   time(&lQuitTime);
   lQuitTime += (time_t) nSeconds;
         while (time(NULL) < lQuitTime)
         ;
   return;
}



static void IntroWindow()
{
   register HWND           hWnd;
   auto    short    nTxtClr  = WHITE | REV_BLUE | HI_INTENSITY;
   auto    short    nBriteClr = YELLOW | REV_BLUE | HI_INTENSITY;

   hWnd = WinCreateWindow(7, 15, 60, 10, nTxtClr,
                          SNGL_LINE_ALL_SIDES, nTxtClr, TRUE);
   WinCenterText(hWnd, 1, "Welcome to ", nTxtClr);
   WinCenterText(hWnd, 3, "Meter Equipment Manufacturing (MEM)", nTxtClr);
   WinCenterText(hWnd, 5, "Variable Area Sizing Program", nTxtClr);
   WinCenterText(hWnd, 7, "by", nBriteClr);
   WinCenterText(hWnd, 8, "Murray Stevenson", nBriteClr);
   Delay(5);
   WinDestroyWindow(hWnd);
   return;
}


static void PhaseWindow()
{
   register HWND    hWnd;
   auto    short    nTxtClr = REV_WHITE | BLUE;
```

```
    int c, end_flag;
    extern int phase;

    hWnd = WinCreateWindow(7, 15, 60, 10, nTxtClr,
                           DBL_LINE_ALL_SIDES, nTxtClr, FALSE);
    WinCenterText(hWnd, 1, "MEM Sizing Program Version 1.00", nTxtClr);
    WinSetCursorPos(hWnd, 3, 5);
    WinTextOut(hWnd,
            "Please type letter of fluid phase;   ",
            nTxtClr);
    WinSetCursorPos(hWnd, 4, 5);
    WinTextOut(hWnd,
            "v for vapor, l for liquid and s for steam:    ",
            nTxtClr);
    ScrCursorOn();
    WinSetCursorPos(hWnd, 4, 48);
    end_flag = FALSE;
    while (!end_flag) {
        c = getch();
        switch (c) {
        case vapor: case Vapor:
                phase = vapor;
                end_flag = TRUE;
                break;
        case liquid: case Liquid:
                phase = liquid;
                end_flag = TRUE;
                break;
        case steam: case Steam:
        phase = steam;
                end_flag = TRUE;
                break;
        }
    }
    WinDestroyWindow(hWnd);
    return;
}
void WinCursorCon(HWND hWnd,char xint, char yint, char xmax, char ymax)
{

    auto    short   nTxtClr = REV_WHITE | BLUE;

    int key, i,n;
    char end_flag, keychar, txt[2], x, y;
    char value[10][6];
    extern float f[10];

    x = 0;
    y = 0;
    end_flag = FALSE;
    txt[1] = '\0';
```

```
for (i=0; i<10; ++i) {
        for (n=0; n<5; ++n)
        value[i][n] = '0';
value[i][n] = '\0';
}


        while (bioskey(1) == 0 && end_flag == FALSE) {
        key = bioskey(0);
        if (isdigit(key & 0xFF) || key == DECIMAL) {
                if (y<=ymax) {
                        keychar = key;
                        txt[0] = keychar;
                        value[x][y] = txt[0];
                        WinTextOut(hWnd,txt,nTxtClr);
                        WinSetCursorPos(hWnd,xint + x,yint + ++y);
                }
        }
        else {
                switch(key) {
                case ARROW_UP:
                        if (x!=0)
                                WinSetCursorPos(hWnd,xint + --x,yint + y);
                        else {
                                x = xmax;
                                WinSetCursorPos(hWnd,xint + x,yint + y);
                        }
                        break;
                case ARROW_RIGHT:
                        if (y<=ymax-1)
                                WinSetCursorPos(hWnd,xint + x,yint + ++y);
                        break;
                case ARROW_LEFT: case BK_SPACE:
                        if (y>=1)
                                WinSetCursorPos(hWnd,xint + x,yint + --y);
                        break;
                case ENTER: case ARROW_DN:
                        if (x!=xmax) {
                                y = 0;
                                WinSetCursorPos(hWnd,xint + ++x,yint + y);
                        }
                        else {
                                x = 0;
                                y = 0;
                                WinSetCursorPos(hWnd,xint + x,yint + y);
                                                }
                        break;
                case PGDN:
                        end_flag = TRUE;
                        break;
        }
```

```
    }
}
    WinSetCursorPos(hWnd,9,3);
    for (i=0; i<=xmax; ++i)
            f[i] = atof(&value[i][0]);
return;
}
void ftoa(float number, char *string)
{

  char fracstr[25] = "";
  double frac = 0.0, ip = 0.0;

  frac = modf((double)number, &ip);
  itoa((int)ip, string, 10);
  strcat(string,".");
  itoa((int)(frac*100.0),fracstr, 10);
  strncat(string,fracstr,1);
  return;
}
static void VapWinInp()
{
    register HWND    hWnd;
    auto    short   nTxtClr = REV_WHITE | BLUE;


    char xint, yint, xmax, ymax;


    hWnd = WinCreateWindow(4, 10, 70, 16, nTxtClr,
                           DBL_LINE_ALL_SIDES, nTxtClr, FALSE);
    WinCenterText(hWnd, 1, "MEM Gas Sizing Program", nTxtClr);
    WinSetCursorPos(hWnd, 3, 5);
    WinTextOut(hWnd,
              "SCFM of gas to be metered at operating conditions  = _____",
              nTxtClr);
    WinSetCursorPos(hWnd, 4, 5);
    WinTextOut(hWnd,
              "Specific gravity of gas at 14.696 psia and 70 F    = _____",
              nTxtClr);
    WinSetCursorPos(hWnd, 5, 5);
    WinTextOut(hWnd,
              "Metering temperature in degrees Rankine          = _____",
              nTxtClr);
              WinSetCursorPos(hWnd, 6, 5);
               WinTextOut(hWnd,
               "Metering pressure in psia (absolute)              = _____",
               nTxtClr);
    WinSetCursorPos(hWnd, 8, 4);
    WinTextOut(hWnd,
              "NOTE: All data entries must contain a decimal point ",
              nTxtClr);
```

```
    WinSetCursorPos(hWnd, 10, 4);
    WinTextOut(hWnd,
               "Right arrow to move right; Left arrow to move left.",
               nTxtClr);
    WinSetCursorPos(hWnd, 11, 4);
    WinTextOut(hWnd,
               "Return or Down arrow to move to beginning of next line.",
               nTxtClr);
    WinSetCursorPos(hWnd, 12, 4);
    WinTextOut(hWnd,
               "Page Down to end data entry.",
               nTxtClr);
               WinSetCursorPos(hWnd, 3, 58);
    xint = 3;
    yint = 58;
    xmax = 3;
    ymax = 4;
    WinCursorCon(hWnd, xint, yint, xmax, ymax);
    WinDestroyWindow(hWnd);
    return;
}

static void VapWinOut()
{
    register HWND    hWnd;
    auto    short   nTxtClr = REV_WHITE | BLUE;
    float QM;
    char QMstring[25] = "";

    ScrCursorOff();
    QM = f[0]*0.465*(sqrt(f[1]*f[2]/f[3]));
    hWnd = WinCreateWindow(7, 15, 60, 10, nTxtClr,
                           DBL_LINE_ALL_SIDES, nTxtClr, FALSE);
    WinCenterText(hWnd, 1, "MEM Gas Sizing Program", nTxtClr);
    WinSetCursorPos(hWnd, 3, 5);
    WinTextOut(hWnd,
               "SCFM DRY AIR @ 100 psig, 70 F (MEM BASE)  QM =",
               nTxtClr);
    ftoa(QM,QMstring);
    WinSetCursorPos(hWnd, 3, 51);
    WinTextOut(hWnd, QMstring,nTxtClr);
          WinSetCursorPos(hWnd, 5, 5);
    WinTextOut(hWnd, "Press any key to continue",nTxtClr);
    getch();
    WinDestroyWindow(hWnd);
    return;
}
static void LiqWinInp()
{
    register HWND    hWnd;
    auto    short   nTxtClr = REV_WHITE | BLUE;
```

```
char xint, yint, xmax, ymax;


hWnd = WinCreateWindow(4, 10, 70, 16, nTxtClr,
                       DBL_LINE_ALL_SIDES, nTxtClr, FALSE);
WinCenterText(hWnd, 1, "MEM Liquid Sizing Program", nTxtClr);
WinSetCursorPos(hWnd, 3, 5);
WinTextOut(hWnd,
        "GPM of application liquid to be metered          = _____",
        nTxtClr);
WinSetCursorPos(hWnd, 4, 5);
WinTextOut(hWnd,
        "Den. of liq. in stand. atm. @ 70 deg. F, #/cubic ft  = _____",
        nTxtClr);
WinSetCursorPos(hWnd, 5, 5);
WinTextOut(hWnd,
        "Density of liquid @ operating conditons, #/cubic ft  = _____",
        nTxtClr);
WinSetCursorPos(hWnd, 6, 5);
        WinTextOut(hWnd,
        "Density of float material, #/cubic ft            = _____",
        nTxtClr);
WinSetCursorPos(hWnd, 7, 5);
        WinTextOut(hWnd,
        "516.6 for brass, 501.1 for S.S. or 217.8 (avg) for PVC/CPVC",
        nTxtClr);
WinSetCursorPos(hWnd, 9, 4);
WinTextOut(hWnd,
        "NOTE: All data entries must contain a decimal point.",
        nTxtClr);
WinSetCursorPos(hWnd, 11, 4);
WinTextOut(hWnd,
        "Right arrow to move right; Left arrow to move left.",
        nTxtClr);
WinSetCursorPos(hWnd, 12, 4);
WinTextOut(hWnd,
        "Return or Down arrow to move to beginning of next line.",
        nTxtClr);
WinSetCursorPos(hWnd, 13, 4);
WinTextOut(hWnd,
        "Page Down to end data entry.",
        nTxtClr);
WinSetCursorPos(hWnd, 3, 61);
xint = 3;
yint = 61;
xmax = 3;
ymax = 4;
WinCursorCon(hWnd, xint, yint, xmax, ymax);
WinDestroyWindow(hWnd);
return;
}
```

```
static void LiqWinOut()
{
   register HWND    hWnd;
   auto    short   nTxtClr = REV_WHITE | BLUE;
   float QM;
   char QMstring[25] = "";

   ScrCursorOff();
   QM = f[0]*(sqrt(f[2]*(f[3]-f[1])/(f[1]*(f[3]-f[2]))));
   hWnd = WinCreateWindow(7, 15, 60, 10, nTxtClr,
                          DBL_LINE_ALL_SIDES, nTxtClr, FALSE);
   WinCenterText(hWnd, 1, "MEM Liquid Sizing Program", nTxtClr);
   WinSetCursorPos(hWnd, 3, 5);
   WinTextOut(hWnd,
            "GPM liquid, Spec. Grav. = 1.00 (MEM BASE) Qm =",
            nTxtClr);
   ftoa(QM,QMstring);
   WinSetCursorPos(hWnd, 3, 52);
   WinTextOut(hWnd, QMstring,nTxtClr);
        WinSetCursorPos(hWnd, 5, 5);
   WinTextOut(hWnd, "Press any key to continue",nTxtClr);
   getch();
   WinDestroyWindow(hWnd);
   return;
}
static void SteWinInp()
{
   register HWND    hWnd;
   auto    short   nTxtClr = REV_WHITE | BLUE;


   char xint, yint, xmax, ymax;


   hWnd = WinCreateWindow(4, 10, 70, 16, nTxtClr,
                          DBL_LINE_ALL_SIDES, nTxtClr, FALSE);
   WinCenterText(hWnd, 1, "MEM Steam Sizing Program", nTxtClr);
   WinSetCursorPos(hWnd, 3, 5);
   WinTextOut(hWnd,
            "Maximum flow of steam, #/hr                    = _____",
            nTxtClr);
   WinSetCursorPos(hWnd, 4, 5);
   WinTextOut(hWnd,
            "Spec. vol. of steam @ operating cond., cubic ft./#   = _____",
            nTxtClr);
   WinSetCursorPos(hWnd, 9, 4);
   WinTextOut(hWnd,
            "NOTE: All data entries must contain a decimal point.",
            nTxtClr);
   WinSetCursorPos(hWnd, 11, 4);
   WinTextOut(hWnd,
            "Right arrow to move right, Left arrow to move left.",
            nTxtClr);
```

```
  WinSetCursorPos(hWnd, 12, 4);
  WinTextOut(hWnd,
             "Return or Down arrow to move to beginning of next line.",
             nTxtClr);
  WinSetCursorPos(hWnd, 13, 4);
  WinTextOut(hWnd,
             "Page Down to end data entry.",
             nTxtClr);
  WinSetCursorPos(hWnd, 3, 61);
  xint = 3;
  yint = 61;
  xmax = 1;
  ymax = 4;
  WinCursorCon(hWnd, xint, yint, xmax, ymax);
  WinDestroyWindow(hWnd);
  return;
}
static void SteWinOut()
{
  register HWND    hWnd;
  auto   short   nTxtClr = REV_WHITE | BLUE;
  float QM;
  char QMstring[25] = "";

  ScrCursorOff();
  QM = f[0]*sqrt(f[1])*0.17;
  hWnd = WinCreateWindow(7, 15, 60, 10, nTxtClr,
                         DBL_LINE_ALL_SIDES, nTxtClr, FALSE);
  WinCenterText(hWnd, 1, "MEM Steam Sizing Program", nTxtClr);
  WinSetCursorPos(hWnd, 3, 5);
  WinTextOut(hWnd,
             "Required MEM meter capacity, SCFM          ",
             nTxtClr);
  ftoa(QM,QMstring);
  WinSetCursorPos(hWnd, 3, 52);
  WinTextOut(hWnd, QMstring,nTxtClr);
      WinSetCursorPos(hWnd, 5, 5);
  WinTextOut(hWnd, "Press any key to continue",nTxtClr);
  getch();
  WinDestroyWindow(hWnd);
  return;
}


main()
{
  auto   HWND    hIntroWnd;
  extern int phase;

  WinInitialize();
  ScrCursorOff();
  IntroWindow();
  PhaseWindow();
```

```
    if (phase == vapor) {
            VapWinInp();
            VapWinOut();
    }
    else if (phase == liquid) {
            LiqWinInp();
            LiqWinOut();
    }
    else {
            SteWinInp();
            SteWinOut();
    }
    WinTerminate();
    ScrCursorOn();
    return(0);
}
```

Appendix C

Viscosity Coupling System Knowledge in PC Plus

This appendix contains the knowledge base for the viscosity coupling system in PC Plus.

```
DOMAIN :: "VISCOSITY COUPLING SYSTEM"
ROOT FRAME :: INPUT


======================
  Global KB data
======================

FRAME STRUCTURE ::
   INPUT
      DECISION
         COLLECTION

KB Files :: (COLLECTION VISCOSIT.k1 DECISION VISCOSIT.k2)
Parameter groups :: (COLLECTION-PARMS DECISION-PARMS INPUT-PARMS)
Rule groups :: (COLLECTION-RULES USER_SELECTED-RULES DECISION-RULES
          WRITE_EXT-RULES UNITS-RULES INPUT-RULES META-RULES )
Number of rules :: 72
Number of meta-rules :: 0
Variables :: (DOMAIN)
TEXTAGS :: ()
Functions :: ()

  ===============
    VARIABLES
  ===============

DOMAIN
  VALUE :: "VISCOSITY COUPLING SYSTEM"


=====================
  Frame :: INPUT
=====================

  IDENTIFIER :: "INPUT-"
  TRANSLATION :: (a coupling system to help you predict the viscosity of
          your mixture. )
  GOALS :: (UNITS1 COMPOSITION LEAVE_FRAME)
  INITIALDATA :: (UNITS)
  PROMPTEVER :: ("This is a coupling system to select a correlation which
          most accurately" :LINE "predicts the viscosity of the vapor
          and/or liquid of a multi or single" :LINE "component system
          and then calculates the viscosity(s) based on this" :LINE "
          correlation.  In addition, this system will determine the
          fluid's state, density and compressibility factor." )
  PARMGROUP :: INPUT-PARMS
  RULEGROUPS :: (INPUT-RULES UNITS-RULES WRITE_EXT-RULES)
  OFFSPRING :: (DECISION)
  INPUT-PARMS  :: (AMMONIA% BENZENE% BITUMEN% CARBON_DIOXIDE%
          CARBON_MONOXIDE% COMPONENTS COMPONENT_INDICES
```

```
        COMPOSITION CYCLO-C6% ETHANE% HELIUM% HYDROGEN%
        HYDROGEN_SULFIDE% ISO-BUTANE% ISO-PENTANE% LEAVE_FRAME
        METHANE% METHANOL% N-BUTANE% N-DECANE% N-HEPTANE%
        N-HEXANE% N-NONANE% N-OCTANE% N-PENTANE% NITROGEN%
        PRESSATM PRESSPSIA PROPANE% TEMPF TEMPK TOLUENE% UNITS
        UNITS1 WATER% )
INPUT-RULES :: (RULE003 RULE004 RULE005 RULE006 RULE007 RULE008 RULE009
        RULE010 RULE026 RULE027 RULE028 RULE029 RULE030 RULE031
        RULE032 RULE033 RULE034 RULE035 RULE036 RULE037 RULE038
        RULE039 RULE040 RULE041 RULE042 RULE043 RULE044 RULE045
        RULE046 RULE047 RULE048 RULE049 RULE050 RULE051 RULE052
        RULE053 RULE054 RULE055 RULE056 RULE057 RULE058 RULE059
        RULE060 RULE061 RULE062 RULE063 RULE064 RULE065 RULE066
        RULE067 )
UNITS-RULES :: (RULE011 RULE012)
WRITE_EXT-RULES :: (RULE013 RULE014 RULE015 RULE016)


=================
  INPUT-PARMS
=================


AMMONIA%
========
 TRANSLATION :: (the mole percent of AMMONIA in your mixture)
 PROMPT :: (Please enter the mole percent of AMMONIA in your mixture.)
 TYPE :: SINGLEVALUED
 EXPECT :: POSITIVE-NUMBER
 USED-BY :: (RULE064)
 RANGE :: (1.e-7 100.0000001)

BENZENE%
========
 TRANSLATION :: (the mole percent of BENZENE in your mixture)
 PROMPT :: (Please enter the mole percent of BENZENE in your mixture.)
 TYPE :: SINGLEVALUED
 EXPECT :: POSITIVE-NUMBER
 USED-BY :: (RULE059)
 RANGE :: (1.e-7 100.0000001)

BITUMEN%
========
 TRANSLATION :: (the mole percent of BITUMEN in your mixture)
 PROMPT :: (Please enter the mole percent of BITUMEN in our mixture.)
 TYPE :: SINGLEVALUED
 EXPECT :: POSITIVE-NUMBER
 USED-BY :: (RULE067)
 RANGE :: (1.e-7 100.0000001)

CARBON_DIOXIDE%
===============
 TRANSLATION :: (the mole percent of CARBON DIOXIDE in your mixture)
```

```
 PROMPT :: ("Please enter the mole percent CARBON DIOXIDE in your
        mixture." )
 TYPE :: SINGLEVALUED
 EXPECT :: POSITIVE-NUMBER
 USED-BY :: (RULE056)
 RANGE :: (1.e-7 100.0000001)

CARBON_MONOXIDE%
================
 TRANSLATION :: (the mole percent of CARBON MONOXIDE in your mixture)
 PROMPT :: ("Please enter the mole percent of CARBON MONOXIDE in your
        mixture." )
 TYPE :: SINGLEVALUED
 EXPECT :: POSITIVE-NUMBER
 USED-BY :: (RULE063)
 RANGE :: (1.e-7 100.0000001)

COMPONENTS
==========
 TRANSLATION :: (the component (s) in your mixture)
 PROMPT :: (Please select the component(s) in your mixture.)
 TYPE :: ASK-ALL
 EXPECT :: (METHANE ETHANE PROPANE ISO-BUTANE N-BUTANE ISO-PENTANE
        N-PENTANE N-HEXANE N-HEPTANE N-OCTANE N-NONANE N-DECANE
        NITROGEN CARBON_DIOXIDE HYDROGEN_SULFIDE TOLUENE BENZENE
        CYCLO-C6 WATER HYDROGEN CARBON_MONOXIDE AMMONIA METHANOL
        HELIUM BITUMEN )
 ANTECEDENT-IN :: (RULE007 RULE008 RULE009 RULE010 RULE026 RULE027
             RULE028 RULE029 RULE030 RULE031 RULE032 RULE033
             RULE034 RULE035 RULE036 RULE037 RULE038 RULE039
             RULE040 RULE041 RULE042 RULE043 RULE044 RULE045
             RULE046 )
 USED-BY :: (RULE003 RULE004 RULE005 RULE006 RULE047 RULE048 RULE049
         RULE050 RULE051 RULE052 RULE053 RULE054 RULE055 RULE056
         RULE057 RULE058 RULE059 RULE060 RULE061 RULE062 RULE063
         RULE064 RULE065 RULE066 RULE067 RULE020 )

COMPONENT_INDICES
=================
 TRANSLATION :: (COMPONENT NUMBER)
 TYPE :: MULTIVALUED
 UPDATED-IN :: (RULE007 RULE008 RULE009 RULE010 RULE026 RULE027 RULE028
           RULE029 RULE030 RULE031 RULE032 RULE033 RULE034 RULE035
           RULE036 RULE037 RULE038 RULE039 RULE040 RULE041 RULE042
           RULE043 RULE044 RULE045 RULE046 )
 ANTECEDENT-IN :: (RULE013)
 DICTIONARY :: INTERNAL

COMPOSITION
===========
 TRANSLATION :: (the component (s) in your mixture)
 TYPE :: MULTIVALUED
 UPDATED-BY :: (RULE003 RULE004 RULE005 RULE006 RULE047 RULE048 RULE049
```

RULE050 RULE051 RULE052 RULE053 RULE054 RULE055 RULE056
RULE057 RULE058 RULE059 RULE060 RULE061 RULE062 RULE063
RULE064 RULE065 RULE066 RULE067)
DICTIONARY :: INTERNAL

CYCLO-C6%
==========
TRANSLATION :: (the mole percent of CYCLO-C6 in your mixture)
PROMPT :: (Please enter the mole percent of CYCLO-C6 in your mixture.)
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE060)
RANGE :: (1.e-7 100.0000001)

ETHANE%
=======
TRANSLATION :: (the mole percent of ETHANE in your mixture)
PROMPT :: (Please enter the mole percent of ETHANE in your mixture.)
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE004)
RANGE :: (1.e-7 100.0000001)

HELIUM%
=======
TRANSLATION :: (the mole percent of HELIUM in your mixture)
PROMPT :: (Please enter the mole percent of HELIUM in your mixture.)
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE066)
RANGE :: (1.e-7 100.0000001)

HYDROGEN%
=========
TRANSLATION :: (the mole percent of HYDROGEN in your mixture)
PROMPT :: (Please enter the mole percent of HYDROGEN in your mixture.)
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE062)
RANGE :: (1.e-7 100.0000001)

HYDROGEN_SULFIDE%
=================
TRANSLATION :: (the mole percent of HYDROGEN SULFIDE in your mixture)
PROMPT :: ("Please enter the mole percent of HYDROGEN SULFIDE in your
mixture.")
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE057)
RANGE :: (1.e-7 100.0000001)

ISO-BUTANE%
===========

```
 TRANSLATION :: (the mole percent of ISO-BUTANE in your mixture)
 PROMPT :: ("Please enter the mole percent of ISO-BUTANE in your mixture."
        )
 TYPE :: SINGLEVALUED
 EXPECT :: POSITIVE-NUMBER
 USED-BY :: (RULE006)
 RANGE :: (1.e-7 100.0000001)

ISO-PENTANE%
============
 TRANSLATION :: (the mole percent of ISO-PENTANE in your mixture)
 PROMPT :: ("Please enter the mole percent of ISO-PENTANE in your
        mixture." )
 TYPE :: SINGLEVALUED
 EXPECT :: POSITIVE-NUMBER
 USED-BY :: (RULE048)
 RANGE :: (1.e-7 100.0000001)

LEAVE_FRAME
===========
 TRANSLATION :: (goal required to leave input frame)
 TYPE :: SINGLEVALUED
 UPDATED-BY :: (RULE016 RULE015)
 DICTIONARY :: INTERNAL

METHANE%
========
 TRANSLATION :: (the mole percent of METHANE in your mixture)
 PROMPT :: (Please enter the mole percent of METHANE in your mixture.)
 TYPE :: SINGLEVALUED
 EXPECT :: POSITIVE-NUMBER
 USED-BY :: (RULE003)
 RANGE :: (1.e-7 100.0000001)

METHANOL%
=========
 TRANSLATION :: (the mole percent of METHANOL in your mixture)
 PROMPT :: (Please enter the mole percent of METHANOL in your mixture.)
 TYPE :: SINGLEVALUED
 EXPECT :: POSITIVE-NUMBER
 USED-BY :: (RULE065)
 RANGE :: (1.e-7 100.0000001)

N-BUTANE%
=========
 TRANSLATION :: (the mole percent of N-BUTANE in your mixture)
 PROMPT :: (Please enter the mole percent of N-BUTANE in your mixture.)
 TYPE :: SINGLEVALUED
 EXPECT :: POSITIVE-NUMBER
 USED-BY :: (RULE047)
 RANGE :: (1.e-7 100.0000001)

N-DECANE%
```

==========
TRANSLATION :: (the mole percent of N-DECANE in your mixture)
PROMPT :: (Please enter the mole percent of N-DECANE in your mixture.)
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE054)
RANGE :: (1.e-7 100.0000001)

N-HEPTANE%
==========
TRANSLATION :: (the mole percent of N-HEPTANE in your mixture)
PROMPT :: (Please enter the mole percent of N-HEPTANE in your mixture.)
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE051)
RANGE :: (1.e-7 100.0000001)

N-HEXANE%
=========
TRANSLATION :: (the mole percent of N-HEXANE in your mixture)
PROMPT :: (Please enter the mole percent of N-HEXANE in your mixture)
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE050)
RANGE :: (1.e-7 100.0000001)

N-NONANE%
=========
TRANSLATION :: (the mole percent of N-NONANE in your mixture)
PROMPT :: (Please enter the mole percent of N-NONANE in your mixture)
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE053)
RANGE :: (1.e-7 100.0000001)

N-OCTANE%
=========
TRANSLATION :: (the mole percent of N-OCTANE in your mixture)
PROMPT :: (Please enter the mole percent of N-OCTANE in your mixture.)
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE052)
RANGE :: (1.e-7 100.0000001)

N-PENTANE%
==========
TRANSLATION :: (the mole percent of N-PENTANE in your mixture)
PROMPT :: (Please enter the mole percent of N-PENTANE in your mixture)
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE049)
RANGE :: (1.e-7 100.0000001)

NITROGEN%
=========
TRANSLATION :: (the mole percent of NITROGEN in your mixture)
PROMPT :: (Please enter the mole percent of NITROGEN in your mixture.)
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE055)
RANGE :: (1.e-7 100.0000001)

PRESSATM
========
TRANSLATION :: (the pressure of your mixture in atmospheres)
PROMPT :: (Please enter the pressure of your mixture in atmospheres.)
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE012 RULE023)
RANGE :: (1.e-7 68166.32)

PRESSPSIA
=========
TRANSLATION :: (the pressure of your mixture in psia)
PROMPT :: (Please enter the pressure of your mixture in psia.)
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE011 RULE022)
RANGE :: (1.e-7 1000000)

PROPANE%
========
TRANSLATION :: (the mole percent of PROPANE in your mixture)
PROMPT :: (Please enter the mole percent of PROPANE in your mixture.)
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE005)
RANGE :: (1.e-7 100.0000001)

TEMPF
=====
TRANSLATION :: (the temperature of your mixture in degrees F)
PROMPT :: (Please enter the temperature of your mixture in degrees F.)
TYPE :: SINGLEVALUED
EXPECT :: NUMBER
USED-BY :: (RULE011)
RANGE :: (-459.67 1000000)

TEMPK
=====
TRANSLATION :: (the temperature of your mixture in degrees K)
PROMPT :: (Please enter the temperature of your mixture in degrees K.)
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE012)
RANGE :: (1.e-7 555810.92)

TOLUENE%
========
TRANSLATION :: (the mole percent of TOLUENE in your mixture)
PROMPT :: (Please enter the mole percent of TOLUENE in your mixture.)
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE058)
RANGE :: (1.e-7 100.0000001)

UNITS
=====
TRANSLATION :: (the system of units you would like to use)
PROMPT :: (Please Select the Desired Systems of Units)
TYPE :: SINGLEVALUED
EXPECT :: (DEGREES_F_AND_PSIA DEGREES_K_AND_ATMOSPHERES)
USED-BY :: (RULE011 RULE012 RULE016 RULE015)
DICTIONARY :: INTERNAL

UNITS1
======
TRANSLATION :: (the units you have chosen to use)
TYPE :: MULTIVALUED
UPDATED-BY :: (RULE011 RULE012)
ANTECEDENT-IN :: (RULE014)
USED-BY :: (RULE022 RULE023)

WATER%
======
TRANSLATION :: (the mole percent of WATER in your mixture)
PROMPT :: (Please enter the mole percent of WATER in your mixture.)
TYPE :: SINGLEVALUED
EXPECT :: POSITIVE-NUMBER
USED-BY :: (RULE061)
RANGE :: (1.e-7 100.0000001)

==========
INPUT-RULES
==========

RULE003
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE013)
If 1) the component s in your mixture is METHANE, and
2) the mole percent of METHANE in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is METHANE,
and
2) send data to an external location.

RULE004
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE003)

If 1) the component s in your mixture is ETHANE, and
2) the mole percent of ETHANE in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is ETHANE,
and
2) send data to an external location.

RULE005
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE004)

If 1) the component s in your mixture is PROPANE, and
2) the mole percent of PROPANE in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is PROPANE,
and
2) send data to an external location.

RULE006
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE005)

If 1) the component s in your mixture is ISO-BUTANE, and
2) the mole percent of ISO-BUTANE in your mixture is less than
100.0000001 but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is
ISO-BUTANE, and
2) send data to an external location.

RULE007
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES

If the component s in your mixture is METHANE,
Then it is definite (100%) that COMPONENT NUMBER is 1.

RULE008
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE007)

If the component s in your mixture is ETHANE,
Then it is definite (100%) that COMPONENT NUMBER is 2.

RULE009
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE008)
If the component s in your mixture is PROPANE,
Then it is definite (100%) that COMPONENT NUMBER is 3.

RULE010
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE009)
If the component s in your mixture is ISO-BUTANE,
Then it is definite (100%) that COMPONENT NUMBER is 4.

RULE026
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE010)
If the component s in your mixture is N-BUTANE,
Then it is definite (100%) that COMPONENT NUMBER is 5.

RULE027
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE026)
If the component s in your mixture is ISO-PENTANE,
Then it is definite (100%) that COMPONENT NUMBER is 6.

RULE028
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE027)
If the component s in your mixture is N-PENTANE,
Then it is definite (100%) that COMPONENT NUMBER is 7.

RULE029
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE028)
If the component s in your mixture is N-HEXANE,
Then it is definite (100%) that COMPONENT NUMBER is 8.

RULE030
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE029)
If the component s in your mixture is N-HEPTANE,
Then it is definite (100%) that COMPONENT NUMBER is 9.

RULE031
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE030)
If the component s in your mixture is N-OCTANE,
Then it is definite (100%) that COMPONENT NUMBER is 10.

RULE032
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE031)
If the component s in your mixture is N-NONANE,
Then it is definite (100%) that COMPONENT NUMBER is 11.

RULE033
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE032)
If the component s in your mixture is N-DECANE,
Then it is definite (100%) that COMPONENT NUMBER is 12.

RULE034
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE033)
If the component s in your mixture is NITROGEN,
Then it is definite (100%) that COMPONENT NUMBER is 13.

RULE035
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE034)
If the component s in your mixture is CARBON_DIOXIDE,
Then it is definite (100%) that COMPONENT NUMBER is 14.

RULE036
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE035)
If the component s in your mixture is HYDROGEN_SULFIDE,
Then it is definite (100%) that COMPONENT NUMBER is 15.

RULE037
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE036)
If the component s in your mixture is TOLUENE,
Then it is definite (100%) that COMPONENT NUMBER is 16.

RULE038
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE037)
If the component s in your mixture is BENZENE,
Then it is definite (100%) that COMPONENT NUMBER is 17.

RULE039
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE038)
If the component s in your mixture is CYCLO-C6,
Then it is definite (100%) that COMPONENT NUMBER is 18.

RULE040
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE039)
If the component s in your mixture is WATER,
Then it is definite (100%) that COMPONENT NUMBER is 19.

RULE041
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE040)
If the component s in your mixture is HYDROGEN,

Then it is definite (100%) that COMPONENT NUMBER is 20.

RULE042
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE041)
If the component s in your mixture is CARBON_MONOXIDE,
Then it is definite (100%) that COMPONENT NUMBER is 21.

RULE043
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE042)
If the component s in your mixture is AMMONIA,
Then it is definite (100%) that COMPONENT NUMBER is 22.

RULE044
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE043)
If the component s in your mixture is METHANOL,
Then it is definite (100%) that COMPONENT NUMBER is 23.

RULE045
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE044)
If the component s in your mixture is HELIUM,
Then it is definite (100%) that COMPONENT NUMBER is 24.

RULE046
=======
SUBJECT :: INPUT-RULES
ANTECEDENT :: YES
DOBEFORE :: (RULE045)
If the component s in your mixture is BITUMEN,
Then it is definite (100%) that COMPONENT NUMBER is 25.

RULE047
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE006)
If 1) the component s in your mixture is N-BUTANE, and

2) the mole percent of N-BUTANE in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is N-BUTANE,
and
2) send data to an external location.

RULE048
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE0047)
If 1) the component s in your mixture is ISO-PENTANE, and
2) the mole percent of ISO-PENTANE in your mixture is less than
100.0000001 but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is
ISO-PENTANE, and
2) send data to an external location.

RULE049
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE048)
If 1) the component s in your mixture is N-PENTANE, and
2) the mole percent of N-PENTANE in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is
N-PENTANE, and
2) send data to an external location.

RULE050
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE049)
If 1) the component s in your mixture is N-HEXANE, and
2) the mole percent of N-HEXANE in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is N-HEXANE,
and
2) send data to an external location.

RULE051
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE050)
If 1) the component s in your mixture is N-HEPTANE, and
2) the mole percent of N-HEPTANE in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is
N-HEPTANE, and
2) send data to an external location.

RULE052
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE051)
If 1) the component s in your mixture is N-OCTANE, and
2) the mole percent of N-OCTANE in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is N-OCTANE,
and
2) send data to an external location, and
3) set the UTILITY of RULE024 to be -100.

RULE053
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE052)
If 1) the component s in your mixture is N-NONANE, and
2) the mole percent of N-NONANE in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is N-NONANE,
and
2) send data to an external location, and
3) set the UTILITY of RULE024 to be -100.

RULE054
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE053)
If 1) the component s in your mixture is N-DECANE, and
2) the mole percent of N-DECANE in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is N-DECANE,
and
2) send data to an external location, and
3) set the UTILITY of RULE024 to be -100.

RULE055
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE054)
If 1) the component s in your mixture is NITROGEN, and
2) the mole percent of NITROGEN in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is NITROGEN,
and
2) send data to an external location.

RULE056
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE055)
If 1) the component s in your mixture is CARBON_DIOXIDE, and
2) the mole percent of CARBON DIOXIDE in your mixture is less than
100.0000001 but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is
CARBON_DIOXIDE, and
2) send data to an external location.

RULE057
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE056)
If 1) the component s in your mixture is HYDROGEN_SULFIDE, and
2) the mole percent of HYDROGEN SULFIDE in your mixture is less than
100.0000001 but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is
HYDROGEN_SULFIDE, and
2) send data to an external location.

RULE058
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE057)
If 1) the component s in your mixture is TOLUENE, and
2) the mole percent of TOLUENE in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is TOLUENE,
and
2) send data to an external location.

RULE059
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE058)
If 1) the component s in your mixture is BENZENE, and
2) the mole percent of BENZENE in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is BENZENE,
and
2) send data to an external location.

RULE060
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE059)
If 1) the component s in your mixture is CYCLO-C6, and

2) the mole percent of CYCLO-C6 in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is CYCLO-C6,
and
2) send data to an external location.

RULE061
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE060)
If 1) the component s in your mixture is WATER, and
2) the mole percent of WATER in your mixture is less than 100.0000001 but
greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is WATER,
and
2) send data to an external location, and
3) set the UTILITY of RULE020 to be 1.

RULE062
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE061)
If 1) the component s in your mixture is HYDROGEN, and
2) the mole percent of HYDROGEN in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is HYDROGEN,
and
2) send data to an external location.

RULE063
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE062)
If 1) the component s in your mixture is CARBON_MONOXIDE, and
2) the mole percent of CARBON MONOXIDE in your mixture is less than
100.0000001 but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is
CARBON_MONOXIDE, and
2) send data to an external location.

RULE064
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE063)
If 1) the component s in your mixture is AMMONIA, and
2) the mole percent of AMMONIA in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is AMMONIA,
and

2) send data to an external location.

RULE065
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE064)
If 1) the component s in your mixture is METHANOL, and
2) the mole percent of METHANOL in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is METHANOL,
and
2) send data to an external location, and
3) set the UTILITY of RULE020 to be 1.

RULE066
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE065)
If 1) the component s in your mixture is HELIUM, and
2) the mole percent of HELIUM in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is HELIUM,
and
2) send data to an external location.

RULE067
=======
SUBJECT :: INPUT-RULES
DOBEFORE :: (RULE066)
If 1) the component s in your mixture is BITUMEN, and
2) the mole percent of BITUMEN in your mixture is less than 100.0000001
but greater than or equal to 1.e-7,
Then 1) it is definite (100%) that the component s in your mixture is BITUMEN,
and
2) send data to an external location.

==========
UNITS-RULES
==========

RULE011
=======
SUBJECT :: UNITS-RULES
If 1) the system of units you would like to use is DEGREES_F_AND_PSIA, and
2) the temperature of your mixture in degrees F is less than 1000001 but
greater than or equal to -459.66, and
3) the pressure of your mixture in psia is less than 1000001 but greater
than or equal to 0,

Then 1) it is definite (100%) that the units you have chosen to use is 1, and
2) set the UTILITY of RULE022 to be -1.

RULE012
=======
SUBJECT :: UNITS-RULES
If 1) the system of units you would like to use is DEGREES_K_AND_ATMOSPHERES,
and
2) the temperature of your mixture in degrees K is less than 555811.48
but greater than or equal to 0, and
3) the pressure of your mixture in atmospheres is less than 68166.39 but
greater than or equal to 0,
Then 1) it is definite (100%) that the units you have chosen to use is 2, and
2) set the UTILITY of RULE023 to be -1.

==========
WRITE_EXT-RULES
==========

RULE013
=======
SUBJECT :: WRITE_EXT-RULES
ANTECEDENT :: YES
If COMPONENT NUMBER is known,
Then send data to an external location.

RULE014
=======
SUBJECT :: WRITE_EXT-RULES
ANTECEDENT :: YES
If the units you have chosen to use is known,
Then send data to an external location.

RULE015
=======
SUBJECT :: WRITE_EXT-RULES
If the system of units you would like to use is DEGREES_F_AND_PSIA,
Then 1) it is definite (100%) that goal required to leave input frame is OKAY,
and
2) inform the user of this decision, and
3) send data to an external location, and
4) (DOS-CALL-RETURN "flash.exe" ""), and
5) instantiate the frame the frame where the viscosity correlation is
selected if appropriate.

RULE016
=======

```
    SUBJECT :: WRITE_EXT-RULES
If  the system of units you would like to use is DEGREES_K_AND_ATMOSPHERES.
Then 1) it is definite (100%) that goal required to leave input frame is OKAY.
       and
     2) inform the user of this decision, and
     3) send data to an external location, and
     4) (DOS-CALL-RETURN "flash.exe" ""), and
     5) instantiate the frame the frame where the viscosity correlation is
        selected if appropriate.


=========================
  Frame :: DECISION
=========================

  IDENTIFIER :: "DECISION-"
  TRANSLATION :: (the frame where the viscosity correlation is selected)
  PARENTS :: (INPUT)
  GOALS :: (VISC_CORRELATION RESULTS_DISPLAY)
  PROMPT1ST :: (PREMISE)
  PROMPT2ND :: (Would you like to select a viscosity correlation?)
  PREMISE :: ($AND
          (KNOWN FRAME COMPOSITION))
  PARMGROUP :: DECISION-PARMS
  RULEGROUPS :: (DECISION-RULES USER_SELECTED-RULES)
  OFFSPRING :: (COLLECTION)
  DECISION-PARMS        ::    (BLANK    CORRELATION_SELECTION    LEAVE_FRAME1
LEAVE_FRAME2
              LINE MOLE%_LIQUID MOLE%_VAPOR NUMBER_OF_COMPONENTS
              RESULTS RESULTS_DISPLAY VISC_CORRELATION
              VISC_CORRELATION_INDICE )
  DECISION-RULES  :: (RULE020 RULE021 RULE022 RULE023 RULE024 RULE025
              RULE068 RULE069 )
  USER_SELECTED-RULES  :: (RULE017 RULE018 RULE019 RULE072)


    ====================
      DECISION-PARMS
    ====================


  BLANK
  =====
   TRANSLATION :: (a parameter used to set the import statement to the
            begin of visc.prn )
   TYPE :: SINGLEVALUED
   UPDATED-IN :: (RULE068 RULE072)
   DICTIONARY :: INTERNAL

  CORRELATION_SELECTION
  =====================
   TRANSLATION :: (user selected viscosity correlation)
```

```
PROMPT :: ("Please select the viscosity correlation you would like to
      use." :LINE "Please select only ONE method." )
TYPE :: ASK-ALL
EXPECT :: (DEAN_AND_STIEL ELY_AND_HANLEY PEDERSEN_AND_FREDENSLUND)
USED-BY :: (RULE017 RULE018 RULE019)
DICTIONARY :: INTERNAL

LEAVE_FRAME1
============
TRANSLATION :: (parameter required to leave decision frame)
TYPE :: SINGLEVALUED
UPDATED-BY :: (RULE022 RULE023 RULE024 RULE020 RULE021 SREFMARK RULE025
      )
ANTECEDENT-IN :: (RULE068)
DICTIONARY :: INTERNAL

LEAVE_FRAME2
============
TRANSLATION :: (parameter required to leave decision frame if user
      selected corr. )
TYPE :: SINGLEVALUED
UPDATED-BY :: (RULE017 RULE018 RULE019)
ANTECEDENT-IN :: (RULE072)
DICTIONARY :: INTERNAL

LINE
====
TRANSLATION :: (a parameter used to display results)
TYPE :: MULTIVALUED
USED-BY :: (RULE070)
CONTAINED-IN :: (RULE069)
DICTIONARY :: INTERNAL

MOLE%_LIQUID
============
TRANSLATION :: (the mole percent of liquid in the mixture)
TYPE :: SINGLEVALUED
USED-BY :: (RULE021)

MOLE%_VAPOR
===========
TRANSLATION :: (the mole percent of vapor in the mixture)
TYPE :: SINGLEVALUED
USED-BY :: (RULE021)

NUMBER_OF_COMPONENTS
====================
TRANSLATION :: (the number of components in the mixture)
TYPE :: SINGLEVALUED
USED-BY :: (RULE022 RULE023 RULE024 RULE021)

RESULTS
=======
```

TRANSLATION :: (a parameter used to help collect results)
TYPE :: MULTIVALUED
UPDATED-BY :: (RULE069 RULE070)
ANTECEDENT-IN :: (RULE071)
USED-BY :: (RULE069)
RANGE :: (0 2)
DICTIONARY :: INTERNAL

RESULTS_DISPLAY
==========
TRANSLATION :: (a goal required to obtain and display results)
TYPE :: SINGLEVALUED
UPDATED-BY :: (RULE069)
DICTIONARY :: INTERNAL

VISC_CORRELATION
==========
TRANSLATION :: (the viscosity correlation selected)
TYPE :: SINGLEVALUED
EXPECT :: (DEAN_AND_STIEL ELY_AND_HANLEY PEDERSEN_AND_FREDENSLUND)
UPDATED-BY :: (RULE022 RULE023 RULE024 RULE017 RULE018 RULE019 RULE020
RULE021 SREFMARK RULE025)
USED-BY :: (SREFMARK RULE025)

VISC_CORRELATION_INDICE
==========
TRANSLATION :: (an integer passed to the external program)
TYPE :: SINGLEVALUED
UPDATED-BY :: (RULE022 RULE023 RULE024 RULE017 RULE018 RULE019 RULE020
RULE021 SREFMARK RULE025)
DICTIONARY :: INTERNAL

==========
DECISION-RULES
==========

RULE020
=======
SUBJECT :: DECISION-RULES
If 1) the component s in your mixture is WATER, or
2) the component s in your mixture is METHANOL,
Then 1) it is definite (100%) that the viscosity correlation selected is
PEDERSEN_AND_FREDENSLUND, and
2) it is definite (100%) that an integer passed to the external program
is 3, and
3) send data to an external location, and
4) it is definite (100%) that parameter required to leave decision frame
is OKAY.

RULE021

=======

SUBJECT :: DECISION-RULES

If 1) retrieve data from an external source, and
2) the mole percent of liquid in the mixture is greater than 1.e-5,

Then 1) it is definite (100%) that the viscosity correlation selected is PEDERSEN_AND_FREDENSLUND, and
2) it is definite (100%) that an integer passed to the external program is 3, and
3) send data to an external location, and
4) it is definite (100%) that parameter required to leave decision frame is OKAY.

RULE022

=======

SUBJECT :: DECISION-RULES

If 1) the number of components in the mixture is less than or equal to 3, and
2) the units you have chosen to use is 1, and
3) the pressure of your mixture in psia is less than or equal to 146.95,

Then 1) it is definite (100%) that the viscosity correlation selected is DEAN_AND_STIEL, and
2) it is definite (100%) that an integer passed to the external program is 1, and
3) send data to an external location, and
4) it is definite (100%) that parameter required to leave decision frame is OKAY.

RULE023

=======

SUBJECT :: DECISION-RULES

If 1) the number of components in the mixture is less than or equal to 3, and
2) the units you have chosen to use is 2, and
3) the pressure of your mixture in atmospheres is less than or equal to 10.,

Then 1) it is definite (100%) that the viscosity correlation selected is DEAN_AND_STIEL, and
2) it is definite (100%) that an integer passed to the external program is 1, and
3) send data to an external location, and
4) it is definite (100%) that parameter required to leave decision frame is OKAY.

RULE024

=======

SUBJECT :: DECISION-RULES
UTILTIY :: -2

If 1) the number of components in the mixture is greater than 0, and
2) the component s in your mixture is not N-OCTANE, and
3) the component s in your mixture is not N-NONANE, and

4) the component s in your mixture is not N-DECANE,
Then 1) it is definite (100%) that the viscosity correlation selected is ELY_AND_HANLEY, and
2) it is definite (100%) that an integer passed to the external program is 2, and
3) send data to an external location, and
4) it is definite (100%) that parameter required to leave decision frame is OKAY.

RULE025
=======
SUBJECT :: DECISION-RULES
UTILITY :: -3
If the viscosity correlation selected is not known with certainty,
Then 1) it is definite (100%) that the viscosity correlation selected is PEDERSEN_AND_FREDENSLUND, and
2) it is definite (100%) that an integer passed to the external program is 3, and
3) send data to an external location, and
4) it is definite (100%) that parameter required to leave decision frame is OKAY.

RULE068
=======
SUBJECT :: DECISION-RULES
ANTECEDENT :: YES
If parameter required to leave decision frame is known,
Then 1) (DOS-CALL-RETURN "visc.exe" ""), and
2) retrieve data from an external source.

RULE069
=======
SUBJECT :: DECISION-RULES
If a parameter used to help collect results is OKAY,
Then 1) it is definite (100%) that a goal required to obtain and display results is OKAY, and
2) inform the user of this decision, and
3) display a parameter used to display results.

==========
USER_SELECTED-RULES
==========

RULE017
=======
SUBJECT :: USER_SELECTED-RULES
UTILITY :: -100
If user selected viscosity correlation is DEAN_AND_STIEL,

Then 1) it is definite (100%) that the viscosity correlation selected is
DEAN_AND_STIEL, and
2) it is definite (100%) that an integer passed to the external program
is 1, and
3) send data to an external location, and
4) it is definite (100%) that parameter required to leave decision frame
if user selected corr. is OKAY.

RULE018
=======
SUBJECT :: USER_SELECTED-RULES
UTILITY :: -100
If user selected viscosity correlation is ELY_AND_HANLEY,
Then 1) it is definite (100%) that the viscosity correlation selected is
ELY_AND_HANLEY, and
2) it is definite (100%) that an integer passed to the external program
is 2, and
3) send data to an external location, and
4) it is definite (100%) that parameter required to leave decision frame
if user selected corr. is OKAY.

RULE019
=======
SUBJECT :: USER_SELECTED-RULES
UTILITY :: -100
If user selected viscosity correlation is PEDERSEN_AND_FREDENSLUND,
Then 1) it is definite (100%) that the viscosity correlation selected is
PEDERSEN_AND_FREDENSLUND, and
2) it is definite (100%) that an integer passed to the external program
is 3, and
3) send data to an external location, and
4) it is definite (100%) that parameter required to leave decision frame
if user selected corr. is OKAY.

RULE072
=======
SUBJECT :: USER_SELECTED-RULES
ANTECEDENT :: YES
If parameter required to leave decision frame if user selected corr. is
known,
Then 1) inform the user of this decision, and
2) (DOS-CALL-RETURN "visc.exe" ""), and
3) retrieve data from an external source.

==========
Frame :: COLLECTION
==========

```
IDENTIFIER :: "COLLECTION-"
TRANSLATION :: (the frame used collect the results)
PARENTS :: (DECISION INPUT)
PROMPT1ST :: (PREMISE)
PROMPT2ND :: (PREMISE)
PREMISE :: ($AND
        (IMPORT
         ('
           (DOS-FILE-IN "visc.prn" LINE))))
PARMGROUP :: COLLECTION-PARMS
RULEGROUPS :: (COLLECTION-RULES)
COLLECTION-PARMS  :: ()
COLLECTION-RULES  :: (RULE070 RULE071)
```

==========
COLLECTION-RULES
==========

RULE070
=======
SUBJECT :: COLLECTION-RULES
If a parameter used to display results is known,
Then it is definite (100%) that a parameter used to help collect results is OKAY.

RULE071
=======
SUBJECT :: COLLECTION-RULES
ANTECEDENT :: YES
If a parameter used to help collect results is OKAY,
Then set the UTILITY of (RULE017 RULE018 RULE019) to be 2.